

8 bits de PODER

Στο AMSTRAD CPC σας



"Οι περιορισμοί δεν αποτελούν πρόβλημα, αλλά πηγή έμπνευσης".

V42_00

Jose Javier García Aranda

INDEX

1 ΓΙΑΤΙ ΝΑ ΠΡΟΓΡΑΜΜΑΤΙΣΕΤΕ ΜΙΑ ΜΗΧΑΝΗ ΤΟΥ 1984 ΣΗΜΕΡΑ;	9
2 ΛΕΙΤΟΥΡΓΙΕΣ 8ΒΡ ΚΑΙ ΧΡΗΣΗ ΜΝΗΜΗΣ	11
2.1 Τι είναι μια βιβλιοθήκη RSX;.....	12
2.2 Λειτουργίες της 8ΒΡ	13
2.3 Αρχιτεκτονική AMSTRAD CPC	14
2.3.1 <i>GOSUB /RETURN Στοίβα</i>	19
2.3.2 <i>Ένα πείραμα για να δείτε τη ROM με το Winape.</i>	20
2.4 Χάρτης μνημής 8ΒΡ και επιλογές συναρμολογησης.....	21
3 ΑΠΑΙΤΟΥΜΕΝΑ ΕΡΓΑΛΕΙΑ.....	25
4 ΤΑ ΠΡΩΤΑ ΒΗΜΑΤΑ ΜΕ 8ΒΡ.....	27
4.1 Εγκαταστήστε το WINAPE	27
4.2 Εξοικειώση με το WINAPE: "HELLO WORLD"	27
4.3 Κατεβάστε τη βιβλιοθήκη 8ΒΡ.....	27
4.4 Εκτελέστε τα DEMO.....	28
4.5 Δημιουργία του πρωτού σας προγράμματος με την 8ΒΡ	29
4.6 Δημιουργήστε το .DSK με το παιχνίδι 8ΒΡ.....	32
5 ΒΗΜΑΤΑ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΑΚΟΛΟΥΘΗΣΕΤΕ ΓΙΑ ΝΑ ΦΤΙΑΞΕΤΕ ΈΝΑ ΠΑΙΧΝΙΔΙ	33
5.1 Δομή καταλογού του έργου σας	33
5.2 Το παιχνίδι σας σε μόνο 3 αρχεία	34
5.3 Δημιουργήστε ένα δίσκο ή μια κασέτα με το παιχνίδι σας	36
5.3.1 <i>Δημιουργία δίσκου</i>	36
5.3.2 <i>Κάνοντας μια κορδέλα με το Winape.</i>	36
5.3.3 <i>Κάντε μια ταινία εύκολα με CPCDiskXP, 2cdt και tape2wav</i>	38
5.3.4 <i>Αντιμετώπιση προβλημάτων LOAD και MEMORY</i>	39
6 ΣΥΝΑΡΜΟΛΟΓΗΣΗ ΒΙΒΛΙΟΘΗΚΗΣ, ΜΟΥΣΙΚΗ ΚΑΙ ΓΡΑΦΙΚΑ.....	42
6.1 MAKE_ALL.ASM.....	43
6.2 Δομή του αρχείου εικόνας	45
6.3 Δομή αρχείου ακολουθίας κινούμενων σχεδίων	45
6.4 Δομή του αρχείου δρομολογησης	46
6.5 Δομή του αρχείου παγκοσμίου χάρτη	46
7 ΚΥΚΛΟΣ ΠΑΙΧΝΙΔΙΟΥ	47
7.1 Πώς να μετρήσετε τα FPS του κύκλου του παιχνιδιού σας	47
8 SPRITES.....	49
8.1 Επεξεργασία Sprites με το SPEDIT και συναρμολογηση Sprites	49
8.2 Εκτύπωση ενός sprite	54
8.3 Αναδιπλωση Sprite	55
8.4 Sprites με αντικατάσταση	57
8.4.1 <i>Χρήση της αντικατάστασης για τη βελτίωση των επικαλύψεων sprite</i>	61
8.4.2 <i>Αντικατάσταση με 4 χρώματα φόντου</i>	62
8.4.3 <i>Αντικατάσταση σε MODE 1</i>	63
8.4.4 <i>Πώς να ζωγραφίσετε sprites "πίσω" από το φόντο</i>	64
8.4.5 <i>Πώς να χρησιμοποιήσετε περισσότερα χρώματα με αντικατάσταση</i>	65

8.5	SPRITES ΜΕ ΕΙΚΟΝΕΣ ΦΟΝΤΟΥ	67
8.6	ΠΙΝΑΚΑΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ SPRITE	69
8.7	ΕΚΤΥΠΩΣΗ ΌΛΩΝ ΤΩΝ SPRITES ΚΑΙ ΤΑΞΙΝΟΜΗΣΗ	74
8.8	ΣΥΓΚΡΟΥΣΕΙΣ ΜΕΤΑΞΥ SPRITES	76
8.9	ΡΥΘΜΙΣΗ ΕΥΑΙΣΘΗΣΙΑΣ ΣΥΓΚΡΟΥΣΗΣ SPRITE	78
8.10	ΠΟΙΟΣ ΣΥΓΚΡΟΥΕΤΑΙ ΚΑΙ ΜΕ ΠΟΙΟΝ: COLSPALL	79
8.10.1	<i>Πώς να προγραμματίσετε ένα multi-shot χρησιμοποιώντας το COLSPALL80</i>	
8.10.2	<i>Ποιος συγκρούεται όταν υπάρχουν πολλές επικαλύψεις</i>	81
8.10.3	<i>Προηγμένη χρήση του byte κατάστασης σε συγκρούσεις</i>	83
8.11	ΠΙΝΑΚΑΣ ΑΚΟΛΟΥΘΙΩΝ ΚΙΝΟΥΜΕΝΩΝ ΣΧΕΔΙΩΝ	84
8.12	ΕΙΔΙΚΕΣ ΑΚΟΛΟΥΘΙΕΣ ΚΙΝΟΥΜΕΝΩΝ ΣΧΕΔΙΩΝ	85
8.12.1	<i>Ακολουθίες θανάτου</i>	86
8.12.2	<i>Ακολουθίες τέλους</i>	87
8.12.3	<i>Αλυσίδες αλληλουχιών</i>	87
8.12.4	<i>Μακρο-ακολουθίες κινούμενων σχεδίων</i>	87
9	ΤΟ ΠΡΩΤΟ ΣΑΣ ΑΠΛΟ ΠΑΙΧΝΙΔΙ.....	91
9.1	ΤΩΡΑ, ΑΣ ΠΗΔΗΕΟΥΜΕ! BOING, BOING!	91
10	ΣΥΝΟΛΑ ΟΘΟΝΗΣ: ΧΑΡΤΕΣ ΔΙΑΤΑΞΗΣ Ή ΠΛΑΚΙΔΙΩΝ.....	95
10.1	ΟΡΙΣΜΟΣ ΚΑΙ ΧΡΗΣΗ ΔΙΑΤΑΞΗΣ.....	95
10.2	ΠΑΡΑΔΕΙΓΜΑ ΠΑΙΧΝΙΔΙΟΥ ΜΕ ΔΙΑΤΑΞΗ	98
10.3	ΠΩΣ ΝΑ ΑΝΟΙΞΕΤΕ ΜΙΑ ΠΥΛΗ ΣΤΗ ΔΙΑΤΑΞΗ.....	100
10.4	ΈΝΑ ΠΑΙΧΝΙΔΙ ΠΑΖΑ: LAYOUT ΜΕ ΦΟΝΤΟ	101
10.5	ΠΩΣ ΝΑ ΕΞΟΙΚΟΝΟΜΗΣΕΤΕ ΜΝΗΜΗ ΣΤΙΣ ΔΙΑΤΑΞΕΙΣ ΣΑΣ	103
11	ΠΡΟΧΩΡΗΜΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΚΑΙ "ΛΟΓΙΚΕΣ ΧΥΜΑ ..	105
11.1	ΜΕΤΡΗΣΗ ΤΗΣ ΤΑΧΥΤΗΤΑΣ ΤΩΝ ΕΝΤΟΛΩΝ	105
11.2	ΔΗΜΙΟΥΡΓΗΣΤΕ ΜΙΑ ΕΝΙΑΤΙΑ ΛΟΓΙΚΗ ΠΟΥ ΘΑ ΔΙΕΠΕΙ ΌΛΕΣ ΤΙΣ ΟΘΟΝΕΣ ΣΑΣ ..	112
11.3	ΤΕΧΝΙΚΗ "MASS LOGIC	113
11.3.1	<i>Μετακινεί 32 sprites με μαζική λογική</i>	114
11.3.2	<i>Εναλλασσόμενη και περιοδική εκτέλεση καταρράκτη</i>	115
11.3.3	<i>Απλό παράδειγμα μαζικής λογικής</i>	117
11.3.4	<i>Μπλόκο" μετακίνησης μοίρας</i>	118
11.3.5	<i>Μαζική λογική τεχνική σε παιχνίδια τύπου "pacman"</i>	119
11.3.6	<i>Μείωση του αριθμού των εντολών σε κύκλο set</i>	121
11.3.7	<i>Διαδρομές που επιταχύνουν το παιχνίδι χειριζόμενες την κατάσταση</i>	125
11.3.8	<i>Δρομολόγηση sprites με "μαζική λογική".</i>	125
12	ΣΥΝΘΕΤΕΣ ΔΙΑΔΡΟΜΕΣ: ΕΝΤΟΛΗ ROUTEALL.....	129
12.1	ΤΟΠΟΘΕΤΕΙ ΈΝΑ SPRITE ΣΤΗ ΜΕΣΗ ΜΙΑΣ ΔΙΑΔΡΟΜΗΣ : ROUTESP	131
12.2	ΔΗΜΙΟΥΡΓΙΑ ΣΥΝΘΕΤΩΝ ΔΙΑΔΡΟΜΩΝ	133
12.2.1	<i>Αναγκαστικές αλλαγές κατάστασης από διαδρομές</i>	133
12.2.2	<i>Αναγκαστικές αλλαγές ακολουθίας από διαδρομές</i>	135
12.2.3	<i>Αναγκαστικές αλλαγές εικόνας από διαδρομές</i>	135
12.2.4	<i>Αναγκαστική αναδρομολόγηση από διαδρομές</i>	139
12.2.5	<i>Αναγκαστικές αλλαγές διαδρομής από τη BASIC</i>	139
12.2.6	<i>Αναγκαστική κίνηση από διαδρομές</i>	141
12.2.7	<i>Πώς να δημιουργήσετε "δυναμικές" (όχι προκαθορισμένες) διαδρομές..</i>	141
12.2.8	<i>Προγραμματισμός διαδρομών, συμπεριλαμβανομένων των μοτίβων</i>	142

12.2.9	<i>Τυπολογία διαδρομής</i>	143
13	ΟΜΑΛΗ ΚΙΝΗΣΗ ΜΙΣΟΥ BYTE	145
14	ΠΑΙΧΝΙΔΙΑ ΚΥΛΙΣΗΣ	147
14.1	STARS: ΚΥΛΙΝΔΡΟΣ ΑΠΟ ΑΣΤΕΡΙΑ Η ΣΤΙΚΤΗ ΓΗ	147
14.2	ΚΥΛΙΣΗ ΜΕ ΧΡΗΣΗ MOVERALL ή/ΚΑΙ AUTOALL	150
14.3	ΤΕΧΝΙΚΗ ΤΟΥ "SPOTTING"	152
14.4	MAP2SP: ΚΥΛΙΣΗ ΜΕ ΒΆΣΗ ΈΝΑΝ ΠΑΓΚΟΣΜΙΟ ΧΑΡΤΗ	155
14.4.1	<i>Χάρτης του κόσμου (Πίνακας χαρτών)</i>	157
14.4.2	<i>Χρήση της λειτουργίας MAP2SP</i>	158
14.4.3	<i>Παράδειγμα αρχείου φάσης</i>	161
14.4.4	<i>Σύγκρουση εχθρού με χάρτη</i>	163
14.4.5	<i>Εικόνες φόντου στην κύλιση σας</i>	164
14.5	ΠΑΡΑΛΛΑΞΗ ΚΥΛΙΣΗΣ	165
14.6	ΔΥΝΑΜΙΚΗ ΕΝΗΜΕΡΩΣΗ ΧΑΡΤΗ: UMAP	166
14.7	KINOΥΜΕΝΑ ΣΧΕΔΙΑ ΚΑΙ ΚΥΛΙΣΗ ΜΕΛΑΝΙΟΥ: ΕΝΤΟΛΗ RINK	168
14.7.1	<i>2D αγωνιστικά αυτοκίνητα</i>	169
14.7.2	<i>Brick Scroll</i>	171
15	ΠΑΙΧΝΙΔΙΑ ΠΛΑΤΦΟΡΜΑΣ	173
16	ΟΡΔΕΣ ΕΧΩΡΩΝ ΣΕ ΠΑΙΧΝΙΔΙΑ ΚΥΛΙΣΗΣ	175
17	ΕΠΑΝΑΠΡΟΣΔΙΟΡΙΣΙΜΟΙ MINI-ΧΑΡΑΚΤΗΡΕΣ: PRINTAT	177
17.1	ΔΗΜΙΟΥΡΓΗΣΤΕ ΤΟ ΔΙΚΟ ΣΑΣ ΑΛΦΑΒΗΤΟ MINI ΧΑΡΑΚΤΗΡΩΝ	178
17.2	ΠΡΟΕΠΙΛΕΓΜΕΝΟ ΑΛΦΑΒΗΤΟ ΓΙΑ ΤΗ ΛΕΙΤΟΥΡΓΙΑ 1	179
18	PSEUDO 3D	181
18.1	3D ΠΡΟΒΟΛΗ	183
18.1.1	<i>Μαθηματικά της ψευδοτρισδιάστατης προβολής</i>	184
18.1.2	<i>Καμπύλες</i>	188
18.2	ΜΕΓΕΘΥΝΣΗ ΕΙΚΟΝΩΝ	189
18.3	ΧΡΗΣΗ ΤΜΗΜΑΤΩΝ	191
19	ΜΟΥΣΙΚΗ	193
19.1	ΕΠΕΞΕΡΓΑΣΙΑ ΜΟΥΣΙΚΗΣ ΜΕ ΤΟΝ ΙΧΝΗΛΑΤΗ WYZ	193
19.2	ΣΥΝΑΡΜΟΛΟΓΗΣΗ ΤΩΝ ΤΡΑΓΟΥΔΙΩΝ	194
19.3	ΤΙ ΝΑ ΚΑΝΕΤΕ ΑΝ ΔΕΝ ΜΠΟΡΕΙΤΕ ΝΑ ΧΩΡΕΣΤΕ ΜΟΥΣΙΚΗ ΣΕ 1400 BYTES	195
20	Σ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΕ 8ΒΡ	197
20.1	ΠΡΩΤΟ ΒΗΜΑ: ΠΡΟΓΡΑΜΜΑΤΙΣΤΕ ΤΟ ΠΑΙΧΝΙΔΙ BASIC	198
20.2	ΒΗΜΑ 2: ΜΕΤΑΦΡΑΣΤΕ ΤΟΝ ΚΥΚΛΟ ΤΟΥ ΠΑΙΧΝΙΔΙΟΥ ΣΑΣ ΑΠΟ BASIC ΣΕ C	199
20.2.1	<i>GOSUB και RETURN στη C</i>	203
20.2.2	<i>Επικοινωνία BASIC προς C με μεταβλητές BASIC</i>	204
20.2.3	<i>Συμβολοσειρές κειμένου BASIC και C</i>	207
20.3	ΤΡΙΤΟ ΒΗΜΑ: ΜΕΤΑΓΛΩΤΤΙΣΗ ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ ΤΟ "COMPILA.BAT"	208
20.4	ΒΗΜΑ 4: ΈΛΕΓΧΟΣ ΤΩΝ ΟΡΙΩΝ ΜΝΗΜΗΣ	210
20.5	ΠΕΜΠΤΟ ΒΗΜΑ: ΕΝΤΟΠΙΣΤΕ ΤΗ ΔΙΕΥΘΥΝΣΗ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ ΠΟΥ ΘΑ ΚΛΗΘΕΙ ΑΠΟ ΤΟ	
BASIC211		
20.6	ΒΗΜΑ 6: ΣΥΜΠΕΡΙΛΑΒΕΤΕ ΣΤΟ ΠΑΙΧΝΙΔΙ .DSK ΤΟ ΝΕΟ ΔΥΑΔΙΚΟ ΑΡΧΕΙΟ	212

20.7	ΑΝΑΦΟΡΑ ΛΕΙΤΟΥΡΓΙΑΣ 8ΒΡ ΣΕ C	213
20.8	ΠΑΡΑΠΟΜΠΗ ΣΕ ΣΥΝΑΡΤΗΣΗ BASIC ΣΕ C ("MINIBASIC")	215
21	ΟΔΗΓΟΣ ΑΝΑΦΟΡΑΣ ΒΙΒΛΙΟΘΗΚΗΣ 8ΒΡ.....	217
21.1	ΛΕΙΤΟΥΡΓΙΕΣ ΒΙΒΛΙΟΘΗΚΗΣ.....	217
21.1.1	3D.....	217
21.1.2	ANIMA	218
21.1.3	ANIMALL.....	219
21.1.4	AUTO	220
21.1.5	AUTOALL	220
21.1.6	COLAY.....	220
21.1.7	COLSP	221
21.1.8	COLSPALL	223
21.1.9	LAYOUT.....	224
21.1.10	LOCATESP	226
21.1.11	MAP2SP.....	226
21.1.12	MOVER.....	228
21.1.13	MOVERALL.....	228
21.1.14	MUSIC	229
21.1.15	PEEK	229
21.1.16	POKE	230
21.1.17	PRINTAT.....	230
21.1.18	PRINTSP.....	231
21.1.19	PRINTSPALL	232
21.1.20	RINK	234
21.1.21	ROUTEALL.....	235
21.1.22	ROUTESP	237
21.1.23	SETLIMITS	238
21.1.24	SETUPSP	238
21.1.25	STARS	240
21.1.26	UMAP	242
22	ΠΩΣ ΝΑ ΦΤΙΑΞΕΤΕ ΈΝΑΝ ΠΙΝΑΚΑ ΑΠΟΤΕΛΕΣΜΑΤΩΝ	243
23	ΠΙΘΑΝΕΣ ΜΕΛΛΟΝΤΙΚΕΣ ΒΕΛΤΙΩΣΕΙΣ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ.....	245
23.1	ΜΝΗΜΗ ΓΙΑ ΤΟΝ ΕΝΤΟΠΙΣΜΟ ΝΕΩΝ ΛΕΙΤΟΥΡΓΙΩΝ.....	245
23.2	ΕΚΤΥΠΩΣΗ ΑΝΑΛΥΣΗΣ PIXEL.....	245
23.3	ΔΙΑΤΑΞΗ ΛΕΙΤΟΥΡΓΙΑΣ 1.....	245
23.4	ΙΚΑΝΟΤΗΤΑ ΚΙΝΗΜΑΤΟΓΡΑΦΗΣΗΣ	245
23.5	ΛΕΙΤΟΥΡΓΙΕΣ ΚΥΛΙΣΗΣ ΥΛΙΚΟΥ	246
23.6	ΜΕΤΑΦΟΡΑ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ 8ΒΡ ΣΕ ΆΛΛΟΥΣ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΕΣ	247
24	ΜΕΡΙΚΑ ΠΑΙΧΝΙΔΙΑ ΠΟΥ ΈΓΙΝΑΝ ΜΕ 8ΒΡ	249
24.1	ΜΕΤΑΛΛΑΓΜΕΝΟΣ ΜΟΝΤΟΥΑ	249
24.2	ΑΝΥΝΝΑΚΙ, ΤΟ ΕΞΩΓΗΙΝΟ ΠΑΡΕΛΘΟΝ ΜΑΣ	250
24.3	NIBIRU	251
24.4	ΦΡΕΣΚΑ ΦΡΟΥΤΑ & ΛΑΧΑΝΙΚΑ.....	252
24.5	"3D RACING ONE.....	252
24.6	ΔΙΑΣΤΗΜΙΚΟ ΦΑΝΤΑΣΜΑ	253
24.7	ΑΙΩΝΙΑ FROGGER.....	254
24.8	ERIDU: Το ΔΙΑΣΤΗΜΙΚΟ ΛΙΜΑΝΙ.....	255
24.9	HAPPY MONTY.....	256

24.10	ΠΙΛΟΤΟΣ BLASTER	256
24.11	NOMWARS.....	257
24.12	PACO, Ο ΑΝΘΡΩΠΟΣ	258
24.13	MINI ΠΑΙΧΝΙΔΙΑ.....	259
24.13.1	<i>Mini-pong</i>	259
24.13.2	<i>Mini-Invaders</i>	260
25	ΠΡΟΣΑΡΤΗΜΑ I: ΟΡΓΑΝΩΣΗ ΤΗΣ ΜΝΗΜΗΣ ΒΙΝΤΕΟ.....	261
25.1	ΤΟ ΑΝΘΡΩΠΙΝΟ ΜΑΤΙ ΚΑΙ Η ΑΝΑΛΥΣΗ ΤΟΥ CPC.....	261
25.2	ΜΝΗΜΗ ΒΙΝΤΕΟ.....	261
25.2.1	<i>Λειτουργία 2</i>	261
25.2.2	<i>Λειτουργία 1</i>	261
25.2.3	<i>Λειτουργία 0</i>	262
25.2.4	<i>Μνήμη οθόνης</i>	262
25.3	ΥΠΟΛΟΓΙΣΜΟΣ ΔΙΕΥΘΥΝΣΗΣ ΟΘΟΝΗΣ.....	264
25.4	ΣΑΡΩΣΕΙΣ ΟΘΟΝΗΣ	264
25.5	ΠΩΣ ΝΑ ΦΤΙΑΞΕΤΕ ΜΙΑ ΟΘΟΝΗ ΦΟΡΤΩΣΗΣ ΓΙΑ ΤΟ ΠΑΙΧΝΙΔΙ ΣΑΣ	265
26	ΠΑΡΑΡΤΗΜΑ II: Η ΠΑΛΕΤΑ	269
27	ΠΡΟΣΑΡΤΗΜΑ III: ΚΩΔΙΚΟΙ ΚΛΕΙΔΙΑ.....	271
29	ΠΡΟΣΑΡΤΗΜΑ IV: ΠΙΝΑΚΑΣ AMSTRAD CPC ASCII.....	273
30	ΠΡΟΣΑΡΤΗΜΑ V: ΟΡΙΣΜΕΝΑ ΗΧΗΤΙΚΑ ΕΦΕ	275
31	ΠΡΟΣΑΡΤΗΜΑ VI: ΕΝΔΙΑΦΕΡΟΥΣΕΣ ΡΟΥΤΙΝΕΣ ΥΛΙΚΟΛΟΓΙΣΜΙΚΟΥ	277
32	ΠΡΟΣΑΡΤΗΜΑ VII: ΠΙΝΑΚΑΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ SPRITE	279
33	ΠΡΟΣΑΡΤΗΜΑ VIII: ΧΑΡΤΗΣ ΜΝΗΜΗΣ 8ΒΡ	281
34	ΠΡΟΣΑΡΤΗΜΑ IX: ΔΙΑΘΕΣΙΜΕΣ ΕΝΤΟΛΕΣ 8ΒΡ	283
35	ΠΡΟΣΑΡΤΗΜΑ X: ΕΠΙΛΟΓΕΣ ΣΥΝΑΡΜΟΛΟΓΗΣΗΣ 8ΒΡ	285
36	ΠΡΟΣΑΡΤΗΜΑ XI: ΑΝΤΙΣΤΟΙΧΙΣΕΙΣ RSX/ΚΛΗΣΕΩΝ	287
37	ΠΡΟΣΑΡΤΗΜΑ XII: ΣΥΝΑΡΤΗΣΕΙΣ 8ΒΡ ΣΕ C	289
38	ΠΡΟΣΑΡΤΗΜΑ XIII: MINIBASIC ΣΕ C	291

1 Γιατί να προγραμματίσετε μια μηχανή του 1984 σήμερα;

Επειδή οι περιορισμοί δεν αποτελούν πρόβλημα αλλά πηγή έμπνευσης.

Οι περιορισμοί, είτε μιας μηχανής είτε ενός ανθρώπου, ή γενικότερα οποιουδήποτε διαθέσιμου πόρου, διεγείρουν τη φαντασία μας για να τους ξεπεράσουμε. Το AMSTRAD, μια μηχανή του 1984 βασισμένη στον μικροεπεξεργαστή Z80, έχει μικρή μνήμη (64KB) και μικρή ικανότητα επεξεργασίας, αλλά μόνο σε σύγκριση με τους σημερινούς υπολογιστές. Αυτή η μηχανή είναι στην πραγματικότητα ένα εκατομμύριο φορές ταχύτερη από εκείνη που κατασκεύασε ο Alan Turing για να αποκρυπτογραφήσει τα μηνύματα της μηχανής enigma το 1944. Όπως όλοι οι υπολογιστές της δεκαετίας του 1980, ο AMSTRAD CPC εκκινήθηκε σε λιγότερο από ένα δευτερόλεπτο, με τον διερμηνέα BASIC έτοιμο να δεχτεί εντολές του χρήστη, καθώς η BASIC ήταν η γλώσσα με την οποία οι προγραμματιστές μάθαιναν και έκαναν τις πρώτες τους εξελίξεις. Το AMSTRAD BASIC ήταν ιδιαίτερα γρήγορο σε σύγκριση με τους ανταγωνιστές του, και αισθητικά ήταν ένας πολύ ελκυστικός υπολογιστής!



Σχ. 1. Το θρυλικό μοντέλο AMSTRAD CPC464

Όσον αφορά τον μικροεπεξεργαστή Z80, δεν μπορεί καν να πολλαπλασιάσει (στη BASIC μπορείτε να πολλαπλασιάσετε, αλλά αυτό βασίζεται σε ένα εσωτερικό πρόγραμμα που υλοποιεί τον πολλαπλασιασμό μέσω προσθέσεων ή μετατοπίσεων καταχωρητών), μπορεί να κάνει μόνο προσθέσεις, αφαιρέσεις και λογικές πράξεις. Παρά ταύτα, ήταν η καλύτερη CPU 8-bit και αποτελούνταν μόνο από 8500 τρανζίστορ, σε αντίθεση με άλλους επεξεργαστές όπως ο M68000 του οποίου το όνομα προέρχεται ακριβώς από το ότι είχε 68000 τρανζίστορ.

CPU	Αριθμός των τρανζίστορ	MIPS (εκατομμύρια εντολές ανά δευτερόλεπτο)	Υπολογιστές και κονσόλες που το ενσωματώνουν
6502	3.500	0,43 @1Mhz	COMMODORE 64, NES, ATARI 800...
Z80	8.500	0,58 @4Mhz	AMSTRAD, COLECOVISION, SPECTRUM, MSX...
Motorola 68000	68.000	2.188 @ 12.5 Mhz	AMIGA, SINCLAIR QL, ATARI ST...
Intel 386DX	275.000	2.1 @16Mhz	PC
Intel 486DX	1.180.000	11 @ 33 Mhz	PC
Pentium	3.100.000	188 @ 100Mhz	PC
ARM1176		4744 @ 1Ghz (1186 ανά πυρήνα)	Raspberry pi 2, Nintendo 3DS, Samsung Galaxy, ...
Intel i7 (5η γενιά)	2.600.000.000	238310 @ 3Ghz (σχεδόν! 500.000 φορές γρηγορότερ α από έναν Z80)	PC
AMD Ryzen 9 3950X (16 πυρήνες)	3.800.000.000	749070 @4.8Ghz (1,3 εκατομμύρια φορές ταχύτερα από τον Z80)	PC

Πίνακας 1Σύγκριση των MIPS

Αυτό καθιστά τον προγραμματισμό εξαιρετικά ενδιαφέροντα και διεγερτικό για την επίτευξη ικανοποιητικών αποτελεσμάτων. Όλος ο προγραμματισμός μας πρέπει να προσανατολίζεται στη μείωση της χωρικής (μνήμη) και χρονικής (πράξεις) υπολογιστικής πολυπλοκότητας, αναγκάζοντάς μας να εφεύρουμε κόλπα, τεχνάσματα, αλγόριθμους κ.λπ. και καθιστώντας τον προγραμματισμό μια συναρπαστική περιπέτεια. Αυτός είναι ο λόγος για τον οποίο ο προγραμματισμός μηχανών με χαμηλή επεξεργαστική ικανότητα είναι μια διαχρονική έννοια, που δεν υπόκειται σε μόδες ούτε εξαρτάται από την εξέλιξη της τεχνολογίας.

Όλος ο κώδικας για αυτό το βιβλίο, συμπεριλαμβανομένης της βιβλιοθήκης για να φτιάξετε τα δικά σας παιχνίδια ή να συνεισφέρετε στη βιβλιοθήκη, βρίσκεται στο έργο GitHub "8BP", σε αυτή τη διεύθυνση URL. Απλά κατεβάστε το zip (σε ένα επόμενο κεφάλαιο θα σας δώσω βήμα προς βήμα)

<https://github.com/jjaranda13/8BP>

Υπάρχει επίσης ένα ιστολόγιο με πολλές πληροφορίες στη διεύθυνση:

<http://8bitsdepoder.blogspot.com.es>

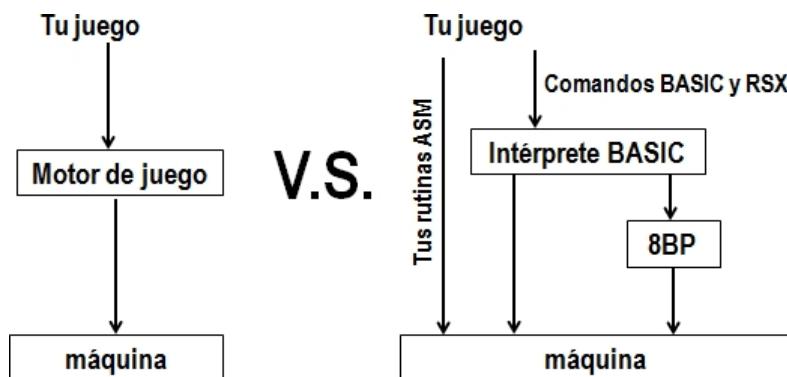
Και ένα κανάλι στο youtube:

https://www.youtube.com/channel/UCThvesOT-jLU_s8a5ZOjBFA

2 Λειτουργίες 8BP και χρήση μνήμης

Η βιβλιοθήκη 8BP δεν είναι μια "μηχανή παιχνιδιού". Είναι κάτι μεταξύ μιας απλής επέκτασης εντολών BASIC και μιας μηχανής παιχνιδιών.

Οι μηχανές παιχνιδιών, όπως το game-maker, το AGD (Arcade Game Designer), το Unity και πολλές άλλες, περιορίζουν σε κάποιο βαθμό τη φαντασία του προγραμματιστή, αναγκάζοντάς τον να χρησιμοποιεί συγκεκριμένες δομές, να προγραμματίζει σε μια περιορισμένη γλώσσα σεναρίων τη λογική ενός εχθρού, να ορίζει και να συνδέει οιθόνες παιχνιδιού κ.λπ.



Σχ. 2 Play Engines έναντι 8BP

Η βιβλιοθήκη 8BP είναι διαφορετική. Είναι μια βιβλιοθήκη ικανή να εκτελεί γρήγορα ό,τι δεν μπορεί να κάνει η BASIC. Πράγματα όπως η εκτύπωση sprites σε πλήρη ταχύτητα, ή η μετακίνηση ομάδων αστεριών στην οιθόνη, είναι πράγματα που δεν μπορεί να κάνει η BASIC, και η 8BP τα κάνει.

Και καταλαμβάνει μόνο 8 KB!

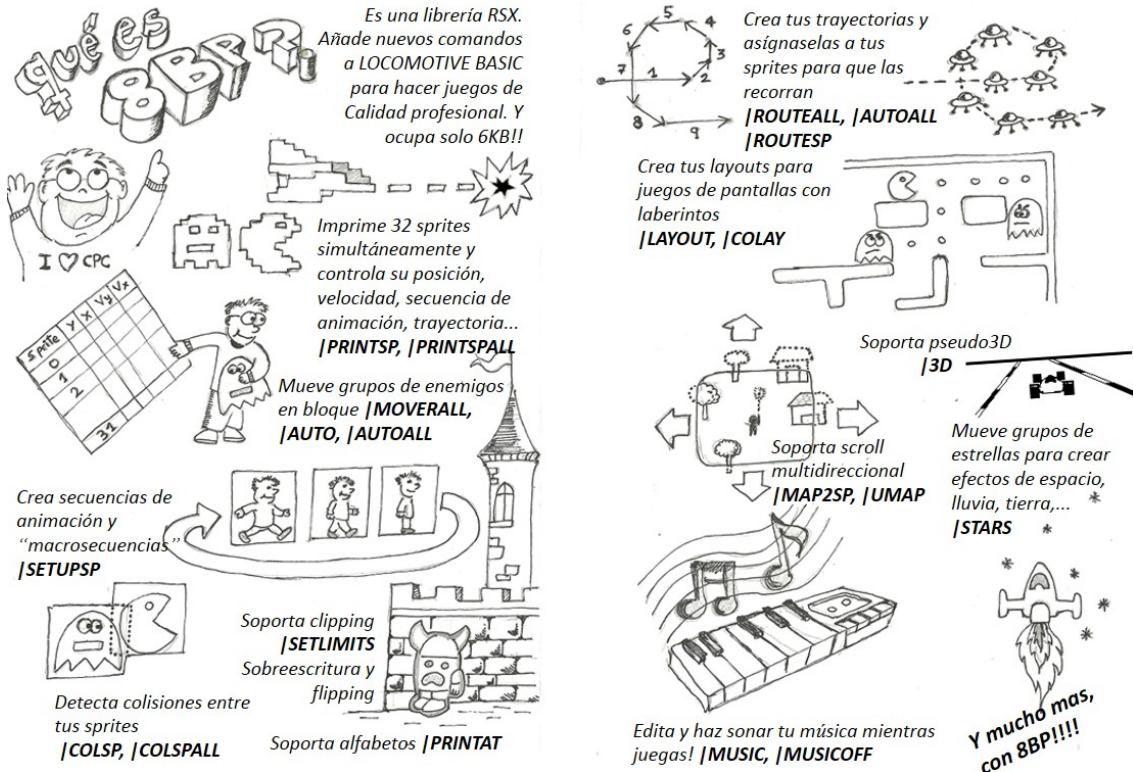
Η BASIC είναι μια διερμηνευόμενη γλώσσα. Αυτό σημαίνει ότι κάθε φορά που ο υπολογιστής εκτελεί μια γραμμή προγράμματος, πρέπει πρώτα να ελέγξει ότι πρόκειται για έγκυρη εντολή συγκρίνοντας τη συμβολοσειρά εντολών με όλες τις έγκυρες συμβολοσειρές εντολών. Στη συνέχεια πρέπει να επικυρώσει συντακτικά την έκφραση, τις παραμέτρους της εντολής και ακόμη και τα επιτρεπόμενα εύρη για τις τιμές αυτών των παραμέτρων. Επιπλέον, διαβάζει τις παραμέτρους σε μορφή κειμένου (ASCII) και πρέπει να τις μετατρέψει σε αριθμητικά δεδομένα. Μόλις γίνει όλη αυτή η εργασία, προχωρά με την εκτέλεση. Λοιπόν, όλη αυτή η εργασία που εκτελείται σε κάθε εντολή είναι αυτό που διαφοροποιεί ένα μεταγλωττισμένο πρόγραμμα από ένα διερμηνευμένο πρόγραμμα, όπως αυτά που έχουν γραφτεί σε BASIC.

Εξοπλίζοντας τη BASIC με τις εντολές που παρέχει η 8BP, είναι δυνατή η δημιουργία παιχνιδιών επαγγελματικής ποιότητας, καθώς η λογική του παιχνιδιού που προγραμματίζετε μπορεί να εκτελεστεί στη BASIC, ενώ οι λειτουργίες που είναι εντατικές για τη CPU, όπως η εκτύπωση στην οιθόνη ή η ανίχνευση συγκρούσεων μεταξύ sprites κ.λπ., εκτελούνται σε κώδικα μηχανής από τη βιβλιοθήκη. Ωστόσο, δεν είναι όλα εύκολα και χωρίς προβλήματα. Παρόλο που η βιβλιοθήκη 8BP θα σας παρέχει πολύ χρήσιμες λειτουργίες στα βιντεοπαιχνίδια, θα πρέπει να τη χρησιμοποιείτε με προσοχή, επειδή κάθε εντολή που καλείτε θα περνάει από το επίπεδο ανάλυσης της BASIC, πριν φτάσει στον υπόκοσμο του κώδικα μηχανής όπου βρίσκεται η λειτουργία, οπότε η απόδοση δεν θα είναι ποτέ βέλτιστη. Θα πρέπει να είστε έξυπνοι και να αποθηκεύετε εντολές, να μετράτε τους χρόνους εκτέλεσης των εντολών και των τιμημάτων του προγράμματός σας και να σκέφτεστε στρατηγικές για την εξοικονόμηση

χρόνου εκτέλεσης. Πρόκειται για μια περιπέτεια εφευρετικότητας και διασκέδασης. Εδώ θα μάθετε πώς να το κάνετε και θα σας παρουσιάσω ακόμη και μια τεχνική που έχω ονομάσει "μαζική λογική", η οποία θα σας επιτρέψει να επιταχύνετε τα παιχνίδια σας σε όρια που ίσως θεωρούσατε αδύνατα.

Εκτός από τη βιβλιοθήκη, έχετε στη διάθεσή σας έναν απλό αλλά πλήρη επεξεργαστή sprite και γραφικών και μια σειρά από θαυμάσια εργαλεία που θα σας επιτρέψουν να απολαύσετε την περιπέτεια του προγραμματισμού ενός μικροϋπολογιστή στον 21ο αιώνα.

Το παρακάτω "ωραίο" σχέδιο παρουσίαζε σχηματικά τις δυνατότητες που παρείχε η 8BP και χρησιμοποιήθηκε σε μια "ρετρό" έκθεση. Σήμερα έχει περισσότερες



δυνατότητες.

Σχ. 3 Σύνοψη της 8BP (έχει επί τον παρόντος περισσότερες εντολές)

2.1 Τι είναι μια βιβλιοθήκη RSX;

RSX είναι το ακρωνύμιο για το Resident System eXtensions. Βιβλιοθήκες όπως η 8BP που παρέχουν εντολές για την επέκταση της BASIC ονομάζονται βιβλιοθήκες RSX.

Στο CPC6128, ορισμένες από τις εντολές που χρησιμοποιούνται για την οδήγηση της μονάδας δίσκου είναι προεγκατεστημένες εντολές RSX, όπως |TAPE, |DISC, |A, |B,

|CPM και άλλα. Αν δεν υπήρχε αυτή η λειτουργικότητα, κάθε ρουτίνα 8BP θα έπρεπε να κληθεί με μια CALL <διεύθυνση>, οπότε η ύπαρξη του RSX κάνει τα προγράμματα πιο κατανοητά.

Δεν είναι όλα ειρήνη και αρμονία. Η χρήση του RSX είναι πιο αργή από την απευθείας χρήση του CALL και επίσης, αν δηλώσουμε 10 νέες εντολές σε μια βιβλιοθήκη, η δέκατη εντολή μπορεί να χρειαστεί 1ms περισσότερο χρόνο για να αρχίσει να εκτελείται από την πρώτη. Η βιβλιοθήκη 8BP έχει 27 εντολές και η τελευταία αρχίζει να εκτελείται 2ms αργότερα επειδή βρίσκεται στο τέλος της λίστας. Αυτό είναι ένα από τα προβλήματα του να είσαι κάτω από τον διερμηνέα BASIC.

Η 8BP αντισταθμίζει αυτό το πρόβλημα τοποθετώντας τις πιο συχνά χρησιμοποιούμενες εντολές στην κορυφή του καταλόγου και αφήνοντας τις λιγότερο συχνά χρησιμοποιούμενες εντολές στο τέλος. Όπως θα διαπιστώσετε σύντομα, η πιο

συχνά χρησιμοποιούμενη εντολή στην 8BP είναι η |PRINTSPALL, η οποία εκτυπώνει όλα τα sprites στην οθόνη. Επομένως, αυτή η εντολή βρίσκεται στην κορυφή της λίστας.

2.2 Λειτουργίες της 8BP

Αφού φορτώσετε τη βιβλιοθήκη με την εντολή: BIN" και καλέστε από τη BASIC τη συνάρτηση _INSTALL_RSX (που ορίζεται στον κώδικα μηχανής) χρησιμοποιώντας την εντολή BASIC:

ΚΛΗΣΗ &6b78

Θα έχετε στη διάθεσή σας τις ακόλουθες εντολές, τις οποίες θα μάθετε να χρησιμοποιείτε με αυτό το βιβλίο

3D, <flag>, #, offsety 3D, 0	Ενεργοποιεί τη λειτουργία ψευδοτρισδιάστατης προβολής.
ANIMA, #	Αλλάζει το πλαίσιο ενός sprite σύμφωνα με την ακολουθία του
ANIMALL	Αλλάζει το καρέ των sprites με ενεργοποιημένη τη σημαία animation (δεν χρειάζεται να την καλέσετε, αρκεί μια σημαία στην εντολή PRINTSPALL για να την καλέσετε).
AUTO, #	Αυτόματη μετακίνηση ενός sprite σύμφωνα με τα Vy,Vx του
AUTOALL, <flag routed>, <flag routed>, <flag routed>, <flag routed>, <flag routed>.	Κίνηση όλων των sprites με ενεργοποιημένη τη σημαία αυτόματης κίνησης
COLAY, threshold_ascii, @collision, # COLAY, @collision, # COLAY, # COLAY	Ανιχνεύει τη σύγκρουση με τη διάταξη και επιστρέφει 1 εάν υπάρχει σύγκρουση. Δέχεται μεταβλητό αριθμό παραμέτρων (πάντα με την ίδια σειρά) από 4 έως καμία.
COLSP, #, @collided%, @collided%, @COLSP, #, @collided%, @collided%. COLSP, 32, ini, τέλος COLSP, 33, @collided% COLSP, 34, dy, dx COLSP, #	Επιστρέφει το πρώτο sprite με το οποίο συγκρούεται το #. Η εντολή μπορεί να ρυθμιστεί με τους κωδικούς 32,33 και 34.
COLSPALL,@who%, @withwho%. COLSPALL, συγκρουστήρας COLSPALL	Επιστρέφει ποιος συγκρούστηκε (collider) και με ποιον συγκρούστηκε (collided).
LAYOUT, y, x, @string\$, @string\$, @string\$, @string\$, @string\$, @string\$, @string\$, @string\$.	Εκτυπώνει λωρίδα εικόνας 8x8 και γεμίζει τη διάταξη χάρτη
LOCATESP, #, y, x	Αλλάζει τις συντεταγμένες ενός sprite (χωρίς να το εκτυπώσει)
MAP2SP, y, x MAP2SP, κατάσταση	Δημιουργεί sprites για να ζωγραφίσει τον κόσμο σε παιχνίδια κύλισης. Τα sprites δημιουργούνται με state = status
MOVER, #, dy, dx	σχετική μετακίνηση ενός μεμονωμένου sprite
OVERALL, dy,dx OVERALL	Σχετική κίνηση όλων των sprites με ενεργή τη σημαία σχετικής κίνησης
MUSIC, C, σημαία, τραγούδι, ταχύτητα MUSIC, σημαία, τραγούδι, ταχύτητα MUSIC	Μια μελωδία αρχίζει να παίζει. Το κανάλι C μπορεί να απενεργοποιηθεί για χρήση με εφέ FX, εάν είναι επιθυμητό Καμία παράμετρος δεν σταματά να ακούγεται.
PEEK, dir, @variable%	Διαβάζει δεδομένα 16bit (μπορεί να είναι αρνητικά)
POKE, dir, value	εισάγετε ένα δεδομένο 16bit (το οποίο μπορεί να είναι αρνητικό)
PRINTAT, σημαία, y, x, @string	Εκτυπώνει μια συμβολοσειρά επαναπροσδιορίσιμων "μίνι-χαρακτήρων".
PRINTSP, #, y, x PRINTSP, # PRINTSP,32, bits	εκτυπώνει ένα μόνο sprite (# είναι ο αριθμός του) ανεξάρτητα από το byte κατάστασης. Εάν έχει καθοριστεί 32, τότε θέτουμε τα bits φόντου
PRINTSPALL, ini, fin, anima, sync PRINTSPALL, ordermode PRINTSPALL	Εκτυπώνει όλα τα sprites με ενεργή σημαία εκτύπωσης. Εάν κληθεί με μία μόνο παράμετρο, ορίζεται η λειτουργία διάταξης.

RINK,tini,color1,color2,...,colorN RINK, άλμα	Περιστρέφει ένα σύνολο μελανιών σύμφωνα με ένα οριζόμενο μοτίβο που αποτελείται από οποιοδήποτε αριθμό μελανιών
ROUTESP, #, βήματα	Σας αναγκάζει να περάσετε N βήματα της διαδρομής του sprite ταυτόχρονα.
ROUTEALL	Τροποποιήστε την ταχύτητα του sprite με τη σημαία διαδρομής (δεν χρειάζεται να την καλέσετε, απλά σημειώστε τη σημαία στο AUTOALL).
SETLIMITS, xmin, xmax, ymin, ymax	Ορίζει το παράθυρο του παιχνιδιού, όπου γίνεται η αποκοπή.
SETUPSP, #, param_number, value SETUPSP, #, 5, Vy, Vx	Τροποποιεί μια παράμετρο ενός sprite. Εάν έχει καθοριστεί η παράμετρος 5, μπορεί να παρέχεται προαιρετικά η Vx.
STARS, initstar, num, color, dy, dx	Κύλιση ενός συνόλου αστεριών
UMAP,adr_ini, adr_end, yini, yfin, xini, xfin	Ενημερώνει τα στοιχεία στον παγκόσμιο χάρτη με ένα υποσύνολο στοιχείων από έναν μεγαλύτερο χάρτη.

Πίνακας 2 Εντολές διαθέσιμες στη βιβλιοθήκη 8BP

Σημειώστε ότι μια κάθετη γραμμή εμφανίζεται στην αρχή κάθε μιας, επειδή πρόκειται για "επεκτάσεις" της BASIC. Ορισμένες μεταβλητές εμφανίζονται με το σύμβολο "%" για να υποδηλώσουν ότι είναι ακέραιοι αριθμοί (όχι δεκαδικοί), αλλά αν χρησιμοποιήσετε το DEFINT για να αναγκάσετε όλες τις μεταβλητές να είναι ακέραιοι αριθμοί, δεν χρειάζεστε το "%".

Επιπλέον, έχετε μια πειραματική εντολή:

| RETROTIME, ημερομηνία

Αυτή η εντολή σας επιτρέπει να μετατρέψετε το CPC σας σε μηχανή του χρόνου, εισάγοντας απλώς την επιθυμητή ημερομηνία-στόχο. Ο μόνος περιορισμός της εντολής είναι ότι πρέπει να εισαγάγετε μια ημερομηνία ίση ή μεταγενέστερη της ημερομηνίας γέννησης του AMSTRAD CPC, τον Απρίλιο του 1984,

| RETROTIME, "01/04/1984".

Παρακαλούμε χρησιμοποιήστε αυτή τη λειτουργία με προσοχή. Θα μπορούσατε να δημιουργήσετε ένα χρονικό παράδοξο και να καταστρέψετε τον κόσμο.

Αν και μπορεί να είστε δύσπιστοι προς το παρόν για το τι μπορείτε να κάνετε με τη βιβλιοθήκη 8BP, σύντομα θα ανακαλύψετε ότι η χρήση αυτής της βιβλιοθήκης σε συνδυασμό με τις προηγμένες τεχνικές προγραμματισμού που θα μάθετε σε αυτό το βιβλίο θα σας επιτρέψει να φτιάξετε επαγγελματικά παιχνίδια στη BASIC, κάτι που ίσως θεωρούσατε αδύνατο.

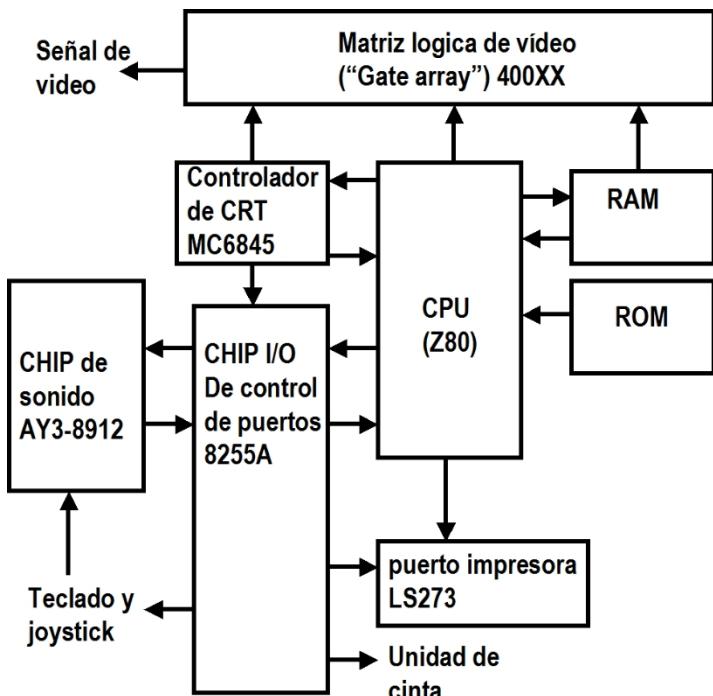
Σημαντική σημείωση για τον προγραμματιστή:

Η βιβλιοθήκη 8BP έχει βελτιστοποιηθεί ώστε να είναι πολύ γρήγορη. Για το λόγο αυτό δεν ελέγχει αν έχετε ορίσει σωστά τις παραμέτρους κάθε εντολής, ούτε αν έχουν τη σωστή τιμή. Εάν κάποια παράμετρος έχει οριστεί λανθασμένα, είναι πολύ πιθανό ο υπολογιστής να κολλήσει κατά την εκτέλεση της εντολής. Ο έλεγχος αυτών των πραγμάτων απαιτεί χρόνο εκτέλεσης και ο χρόνος είναι ένας πόρος που δεν μπορεί να σπαταληθεί, ούτε ένα χλιοστό του δευτερολέπτου.

2.3 Αρχιτεκτονική AMSTRAD CPC

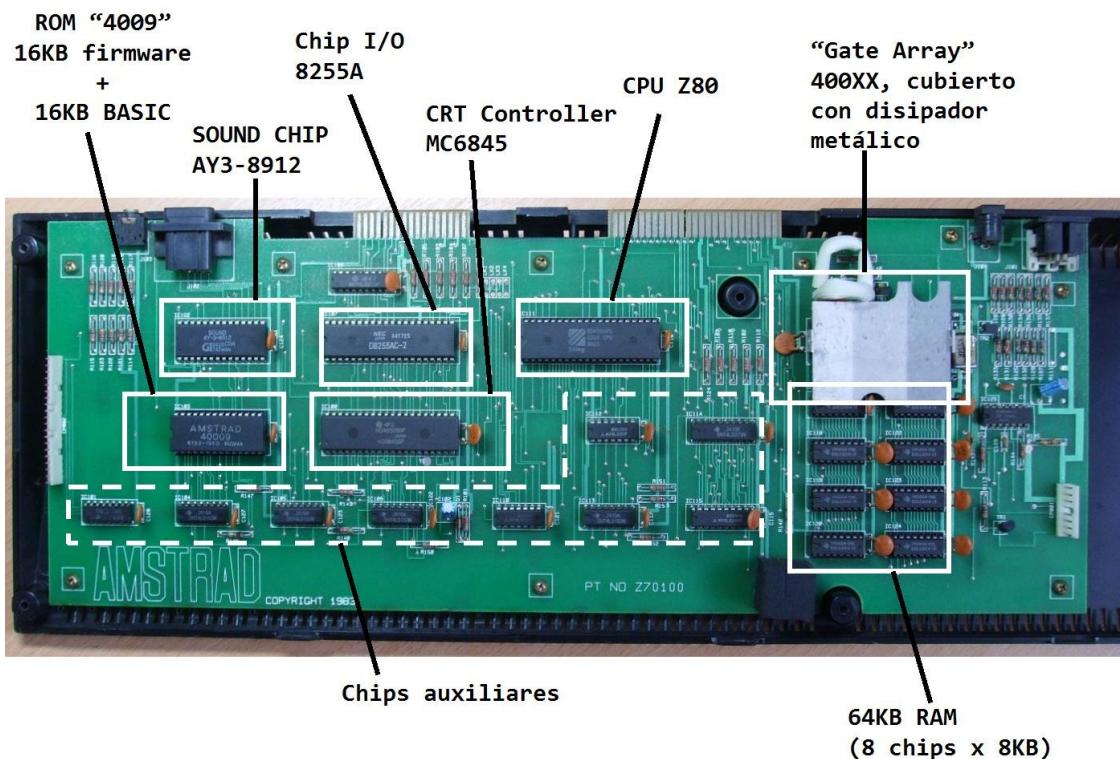
Αυτή η ενότητα είναι χρήσιμη για να κατανοήσετε αργότερα πώς η βιβλιοθήκη 8BP χρησιμοποιεί τη μνήμη.

Το AMSTRAD είναι ένας υπολογιστής βασισμένος στον μικροεπεξεργαστή Z80, που λειτουργεί στα 4MHz. Όπως φαίνεται στο διάγραμμα αρχιτεκτονικής του, τόσο η CPU όσο και η λογική συστοιχία βίντεο (που ονομάζεται "gate array") έχουν πρόσβαση στη μνήμη RAM, οπότε για να "εναλλάσσονται", οι προσπελάσεις μνήμης από την CPU καθυστερούν, με αποτέλεσμα η πραγματική ταχύτητα να είναι 3,3Mhz. Αυτό εξακολουθεί να είναι αρκετά μεγάλη ισχύς.



Σχ. 4 Αρχιτεκτονική του AMSTRAD

Η μνήμη RAM βίντεο προσπελαύνεται από το τσιπ gate array 50 φορές ανά δευτερόλεπτο προκειμένου να σταλεί μια εικόνα στην οθόνη. Σε παλαιότερους υπολογιστές (όπως ο Sinclair ZX81) η εργασία αυτή ανατέθηκε στον επεξεργαστή, αφαιρώντας του ακόμα περισσότερη ισχύ.



Σχ. 5 Προσδιορισμός των εξαρτημάτων στην πλακέτα

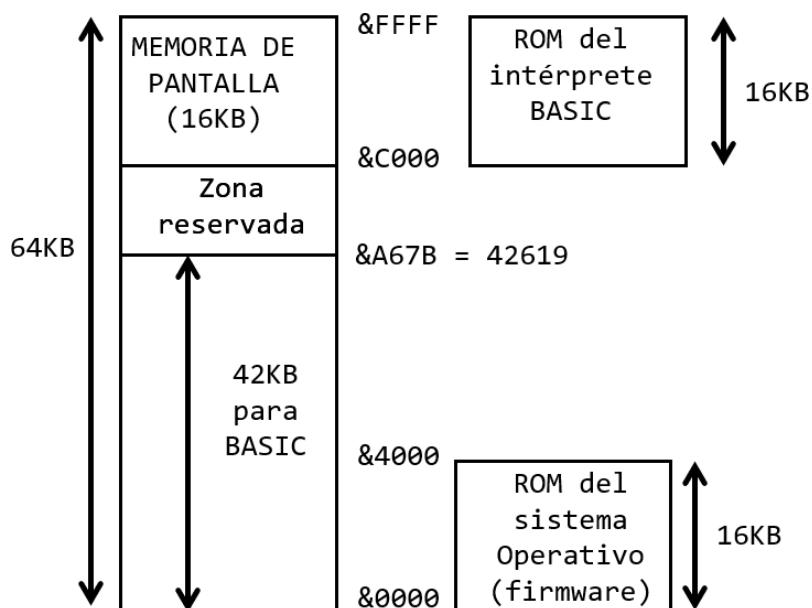
Η συστοιχία πυλών είναι ένα τσιπ που περιέχει πολλές λογικές πύλες, σχεδιασμένες ειδικά για το AMSTRAD. Στο ZX Spectrum υπάρχει ένα παρόμοιο chip που ονομάζεται ULA (Uncommitted Logic Array - δεν πρέπει να συγχέεται με την ALU). Τόσο το amstrad όσο και το ZX είναι τσιπ που έχουν σχεδιαστεί αποκλειστικά για

αυτούς τους υπολογιστές. Στον ZX, εκτός από το ότι χρησιμοποιείται για τη δημιουργία

η εικόνα βίντεο χρησιμοποιούνταν επίσης για την ανάγνωση του πληκτρολογίου και την είσοδο της κασέτας, ωστόσο, στο AMSTRAD αυτές οι λειτουργίες εκτελούνται από άλλα τσιπ, όπως το 8255A.

Ο Z80 διαθέτει δίαυλο διευθύνσεων 16bit, οπότε δεν μπορεί να διευθυνθεί σε περισσότερα από 64KB. Ωστόσο, το Amstrad διαθέτει 64kB RAM και 32kB ROM. Προκειμένου να τις διευθυνθεί σε περισσότερα από 64KB, η μετατροπή των τραπεζών, έτσι ώστε, για παράδειγμα, αν κληθεί μια εντολή BASIC, να μετατρέψει στην τράπεζα ROM όπου είναι αποθηκευμένος ο διερμηνέας BASIC, η οποία επικαλύπτεται με τα 16KB του χώρου της οθόνης. Αυτός ο μηχανισμός είναι απλός και αποτελεσματικός.

Εκτός από τη ROM που περιέχει τον διερμηνέα BASIC των 16KB που βρίσκεται στην περιοχή της υψηλής μνήμης, υπάρχουν άλλα 16KB ROM που βρίσκονται στη χαμηλή μνήμη, όπου βρίσκονται οι ρουτίνες firmware (αυτό που θα μπορούσε να θεωρηθεί το λειτουργικό σύστημα αυτού του μηχανήματος). Συνολικά (διερμηνέας BASIC και υλικολογισμικό) ανέρχονται σε 32KB.



Σχ. 6 Μνήμη AMSTRAD

Όπως φαίνεται στο χάρτη μνήμης, από τα 64KB της μνήμης RAM, τα 16KB (από το &C000 έως το &FFFF) είναι μνήμη βίντεο. Τα προγράμματα BASIC μπορούν να καταλαμβάνουν από τη θέση &40 (διεύθυνση 64) έως 42619, διότι πέρα από αυτό υπάρχουν μεταβλητές συστήματος. Αυτό σημαίνει ότι περίπου 42KB είναι διαθέσιμα για τη BASIC, όπως μπορούμε να δούμε εκτυπώνοντας τη μεταβλητή συστήματος HIMEM (συντομογραφία του "High Memory").

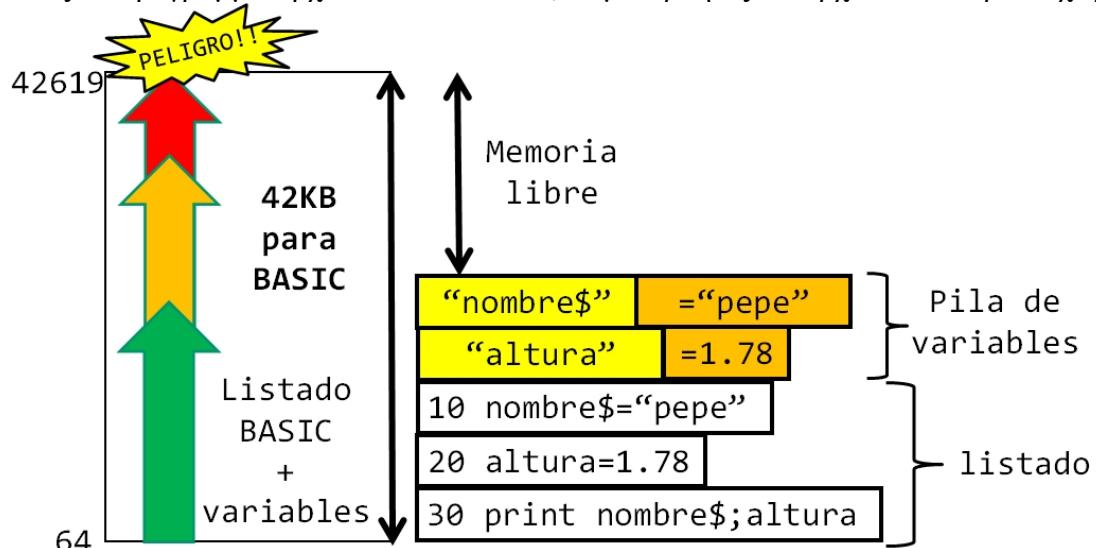
```
print HIMEM
42619
Ready
```

Σχ. 7 Μεταβλητή συστήματος HIMEM

Η λειτουργία της BASIC λαμβάνει υπόψη την αποθήκευση του προγράμματος σε αυξανόμενες διευθύνσεις από τη θέση &40. Μόλις ξεκινήσει η εκτέλεση, οι μεταβλητές που δηλώνονται πρέπει να καταλάβουν χώρο για να αποθηκεύσουν τις τιμές που παίρνουν και επειδή δεν μπορούν να καταλάβουν την ίδια περιοχή όπου αποθηκεύεται

το πρόγραμμα, απλά αρχίζουν να αποθηκεύονται πάνω από την τελευταία διεύθυνση που καταλαμβάνει η λίστα BASIC.

Στο AMSTRAD, κάθε μεταβλητή καταλαμβάνει πληροφορίες με το όνομα και την τιμή της. Μια αριθμητική μεταβλητή τύπου integer θα καταλαμβάνει 2 bytes μνήμης για την τιμή, αλλά καταλαμβάνει άλλα τόσα bytes για την αποθήκευση του ονόματος. Οι πραγματικές μεταβλητές (με δεκαδικά ψηφία) καταλαμβάνουν 5 bytes. Κάθε φορά που χρησιμοποιούμε μια μεταβλητή θα καταναλώνουμε μνήμη ξεκινώντας από την πρώτη ελεύθερη διεύθυνση πάνω από την καταχώριση της BASIC. **Υπάρχει ο κίνδυνος ότι αν δημιουργήσουμε πολλές μεταβλητές και η λίστα BASIC είναι πολύ μεγάλη, η στοίβα των μεταβλητών μπορεί να φάει όλη την ελεύθερη μνήμη.** Σε αυτή την περίπτωση, το πρόγραμμα BASIC θα καταστραφεί και θα σταματήσει να λειτουργεί. Μόλις το πρόγραμμα αρχίσει να εκτελείται, οι μεταβλητές θα αρχίσουν να τρώνε χώρο



και όταν γεμίσουν όλη τη διαθέσιμη μνήμη RAM, θα δώσει σφάλμα πλήρους μνήμης.

Σχήμα 8 Αύξηση της κατανάλωσης μνήμης για λίστες και μεταβλητές BASIC

Μπορείτε να παρακολουθείτε ανά πάσα στιγμή πόση μνήμη έχετε στη διάθεσή σας με την εντολή **FRE(0)**, την τιμή της οποίας μπορείτε να εκτυπώσετε ή να φορτώσετε σε μια μεταβλητή. Μπορείτε να κάνετε αυτό το πείραμα, θα δείτε πώς καταφέρνετε να καταναλώσετε όλη τη μνήμη του CPC με έναν πίνακα ακεραίων αριθμών:

```

print fre(0)
42209
Ready
list
10 CLEAR:DEFINT a-z
20 DIM a(21099)
30 PRINT FRE(0)
Ready
run
0
Ready
■

```

Αυτό το απλό πρόγραμμα καταλαμβάνει όλη τη μνήμη μεταξύ του αθροίσματος της λίστας BASIC και του πίνακα.

Παρατηρήστε πώς η FRE(0) μας λέει ότι δεν έχει απομείνει ούτε ένα byte.

Κάθε φορά που εκχωρείται μια τιμή σε μια κυριολεκτική μεταβλητή, όπως η name\$, η μεταβλητή θα μετατοπίζεται, αφήνοντας όλο και λιγότερο διαθέσιμο χώρο, αν και ο χώρος που προηγουμένως καταλαμβάνει θα επισημαίνεται ως "διαθέσιμος". Όταν ένα πρόγραμμα ξεμείνει από μνήμη, θα συμπιέσει όλες τις μεταβλητές, απελευθερώνοντας όλο τον "διαθέσιμο" χώρο. Αυτό το αποτέλεσμα επιτυγχάνεται επίσης με την εκτέλεση της **FRE("")**. Άλλα προσέξτε, η Amstrad BASIC δεν σας επιτρέπει να εκτελέσετε απλά την **FRE("")**, γιατί θα σας δώσει ένα "συντακτικό σφάλμα". Πρέπει τουλάχιστον να αναθέσετε μια μεταβλητή, για παράδειγμα **p=FRE("")**. Αυτή η εντολή εκτελεί αυτό που σήμερα είναι γνωστό σε ορισμένες

γλώσσες ως "συλλογή σκουπιδιών".

Αν κατά τη διάρκεια της εκτέλεσης ενός προγράμματος έχετε πρόβλημα γεμάτης μνήμης, ίσως μπορέσετε να το λύσετε απλά εκτελώντας περιοδικά μια γραμμή όπως αυτή:

100 c=c+1

**110 αν c και 63 =0 τότε 120 αλλιώς p=fre("") :FRE κάθε 64 κύκλους
120 το πρόγραμμα συνεχίζεται**

Αυτή η λύση λειτουργεί μόνο αν το πρόβλημα οφείλεται σε κάποια εντολή που καταναλώνει λίγο περισσότερο από αυτό που είναι ελεύθερο λίγο πριν την αυτόματη εκτέλεση της FRE. Αν είναι αυτή η περίπτωση, αυτή η λύση θα λειτουργήσει. Ωστόσο, είναι πολύ "σπάνιο" να λυθεί με αυτόν τον τρόπο, διότι αν το Amstrad δεν έχει μνήμη, κάνει αυτόματα FRE και θεωρητικά δεν χρειάζεται να το κάνει. Μια άλλη απλούστερη (και πιο αποτελεσματική) λύση είναι να διαγράψετε ή να συντομεύσετε τις γραμμές REM, για να δώσετε λίγη περισσότερη ελεύθερη μνήμη στο Amstrad. Αν υπάρχει αρκετή μνήμη και εξακολουθείτε να έχετε μια μνήμη γεμάτη, πρόκειται για πρόβλημα υπερβολικού GOSUB χωρίς RETURN.

Κάθε αριθμητική μεταβλητή καταναλώνει την ακόλουθη μνήμη:

- το όνομα της μεταβλητής: N bytes
- η τιμή, με το τελευταίο byte να μηδενίζεται, υποδεικνύοντας το τέλος της κυριολεξίας: 2 bytes

Κάθε μεταβλητή καταναλώνει την ακόλουθη μνήμη:

- το όνομα της μεταβλητής: N bytes
- διεύθυνση μνήμης (ή "δείκτης") όπου αρχίζει η τιμή του: 2 bytes
- η τιμή, με το τελευταίο byte να μηδενίζεται, υποδεικνύοντας το τέλος του κυρίου: N bytes

Κάθε φορά που μια κυριολεκτική μεταβλητή μετατοπίζεται αναθέτοντάς της μια νέα τιμή, η νέα τιμή φορτώνεται στη διαθέσιμη περιοχή μνήμης και ο δείκτης επαναπροσδιορίζεται ώστε να δείχνει στη νέα διεύθυνση, αφήνοντας την παλιά τιμή χωρίς δείκτη από οποιαδήποτε μεταβλητή. Αυτός ο προηγούμενος χώρος ή το "κενό" είναι διαθέσιμος, αλλά απαιτείται μια εκκαθάριση για να συμπιεστούν όλες οι μεταβλητές και να απελευθερωθούν όλα τα διαθέσιμα κενά.

Ας δούμε ένα απλό παράδειγμα που σας δείχνει τον διαθέσιμο χώρο κατά τη μετεγκατάσταση μιας μεταβλητής literal (ή "string"), που προφανώς ξοδεύει όλο και περισσότερη μνήμη, παρόλο που η αχρησιμοποίητη μνήμη παραμένει διαθέσιμη (πρόκειται για "τρύπες" στη στοίβα μεταβλητών). Αν προσπαθήσετε να κάνετε το ίδιο με μια αριθμητική μεταβλητή, η κατανάλωση μνήμης θα είναι σταθερή, επειδή οι αριθμητικές μεταβλητές καταλαμβάνουν πάντα τον ίδιο χώρο και δεν μετατοπίζονται όταν αλλάζουν τιμή, ενώ οι κυριολεκτικές μεταβλητές (ή "συμβολοσειρές") αλλάζουν μήκος όταν αλλάζουν τιμή.

```

10 number=rnd*100
20 c$=str$(number)
30 print fre(0)
40 goto 10

```

Όπως μπορείτε να δείτε, σε κάθε επανάληψη υπάρχει λιγότερη διαθέσιμη μνήμη, αλλά αν το αφήσετε να εκτελεστεί επ' αόριστον, όταν εξαντληθεί η μνήμη, το AMSTRAD θα εκτελέσει μια διαδικασία για να καθαρίσει την αχρησιμοποίητη μνήμη και θα έχει ξανά μνήμη. Αυτή η διαδικασία είναι η ίδια με εκείνη που εκτελείτε την FRE(" ")�.

Δεν πρέπει να συγχέεται με την εντολή CLEAR η οποία ελευθερώνει χώρο αφαιρώντας όλες τις μεταβλητές από τη στοίβα μεταβλητών.

Είναι σκόπιμο να εκτελείτε την FRE(" ") περιοδικά στο πρόγραμμά σας, καθώς η λειτουργία της συμπίεσης όλων των μεταβλητών όταν έχει καταναλωθεί όλη η μνήμη είναι περισσότερη δουλειά (και επομένως πιο αργή) από ότι αν εκτελείτε περιοδικά μια FRE(" ") που έχει λίγη δουλειά να κάνει.

Αν το πρόγραμμά σας παράγει ένα σφάλμα πλήρους μνήμης μετά από λίγο, δοκιμάστε να εκτελείτε περιοδικά ένα FRE("")�, διαγράψτε γραμμές "REM" για να του δώσετε περισσότερη ελεύθερη μνήμη ή ελέγχτε ότι δεν πρόκειται για περίσσεια. φωλιασμός του GOSUB, το οποίο είναι το πιο συχνό σφάλμα.

Ready run
42150
42137
42124
42111
42098
42086
42073
42060
42047
42034
42021
42008
41995
41982
41969
41956
41943
41931
41918

<μετά από πολλές επαναλήψεις>

141
128
115
102
89
76
64
51
38
25
12
42137
42124
42111
42098
42085
42072
42060

Τέλος, μια περιέργεια: το CPC 464 σας δίνει μια FRE(0) 43.553 bytes ελεύθερη ενώ το 6128 σας δίνει λιγότερη μνήμη, συγκεκριμένα 42.249 bytes. Αυτό σχετίζεται άμεσα με το γεγονός ότι αν κάνετε μια PRINT HIMEM στο CPC464 παίρνετε 43903, ενώ στο CPC6128 παίρνετε 42619.

2.3.1 GOSUB /RETURN Στοίβα

Κάθε φορά που εκτελείτε το GOSUB στο Amstrad BASIC, το σύστημα πρέπει να δείχνει πού πρέπει να επιστρέψει όταν κάνετε RETURN. Λοιπόν, το Amstrad CPC διαθέτει μια στοίβα 83 θέσεων για την αποθήκευση των αλμάτων. Είναι σύνηθες να κάνετε κάποιο λάθος στον προγραμματισμό και σε κάποιο IF του υποπρογράμματος να επιστρέφετε με ένα GOTO (δηλαδή να μην επιστρέφετε) οπότε συσσωρεύονται αν δεν είστε προσεκτικοί και μπορεί να λάβετε ένα σφάλμα "memory full" κατά την εκτέλεση του προγράμματός σας.

<pre> 10 GOSUB 100 20 REM εδώ δεν φτάνει ποτέ 100 i=i+1 110 PRINT i 120 GOTO 10 </pre>	73 74 75 76 77 78 79 80 81 82 83 Memory full in 100 Ready
---	---

Σε αυτό το παράδειγμα, ακόμη και αν εκτυπώσετε την υπόλοιπη μνήμη αντικαθιστώντας τη γραμμή 110 με :

110 print i: print fre(0)

Θα δείτε ότι η διαθέσιμη μνήμη δεν μειώνεται και παρόλο που έχετε σχεδόν όλη τη μνήμη ελεύθερη, το Amstrad δίνει memory full. Αυτή η περίπτωση οφείλεται στην εμπλοκή **GOSUB**, όχι επειδή δεν υπάρχει ελεύθερη μνήμη. **Μόνο 83 GOSUB επιτρέπονται χωρίς RETURN.**

Δεν μπορώ να σας πω με βεβαιότητα ποια είναι η περιοχή μνήμης όπου ο διερμηνέας Amstrad BASIC αποθηκεύει τις 83 διευθύνσεις για το RETURN, αλλά πιθανώς βρίσκεται στην περιοχή μνήμης του συστήματος (6KB από 42619 έως 49152).

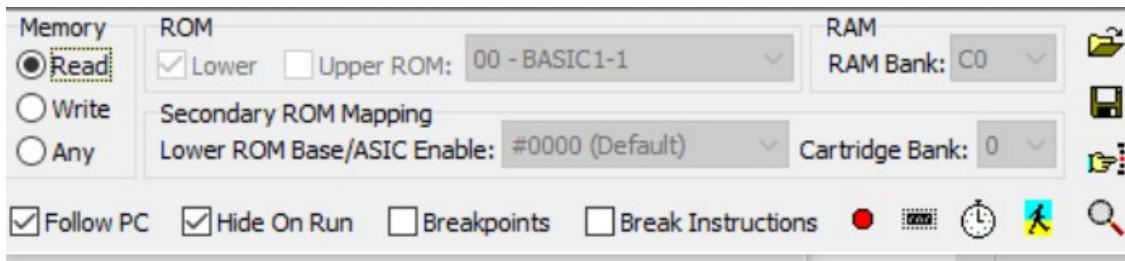
Πρόκειται για μια περιοχή 6KB ακριβώς πριν από τη μνήμη οθόνης (49152 είναι &C0000) όπου η BASIC αποθηκεύει τους επανεγγράψιμους χαρακτήρες, καθώς και τη "στοίβα". Πρόκειται για μια περιοχή που χρησιμοποιείται για την αποθήκευση της διεύθυνσης μνήμης στην οποία βρίσκεται ένα πρόγραμμα πριν μεταβεί σε μια ρουτίνα, ώστε να μπορεί να επιστρέψει στο σημείο που βρισκόταν όταν τελειώνει η ρουτίνα. Χρησιμοποιείται σε χαμηλό επίπεδο (γλώσσα συναρμολόγησης). Η στοίβα αυξάνεται σε χαμηλότερες διευθύνσεις καθώς αποθηκεύει διευθύνσεις (δηλαδή ξεκινάει από την 49152 και παίρνει όλο και μικρότερες διευθύνσεις) και όταν επιστρέφει από μια ρουτίνα μειώνεται και πάλι. Μπορεί να σκεφτείτε ότι, αν μια ρουτίνα καλεί μια άλλη ρουτίνα και αυτή καλεί μια άλλη ρουτίνα κ.ο.κ., η στοίβα θα μεγάλωνε τόσο πολύ που θα έφευγε από την περιοχή "σύστημα" και θα εισέβαλε σε άλλες περιοχές, αλλά μην ανησυχείτε, το 8BP διατηρεί τη στοίβα υπό έλεγχο και το ίδιο κάνει και το Amstrad BASIC.

2.3.2 Ένα πείραμα για να δείτε τη ROM με το Winape

Μπορείτε να χρησιμοποιήσετε το Winape για να δείτε τι περιέχει η ROM του διερμηνέα BASIC που επικαλύπτεται από τις διευθύνσεις της οθόνης, και το ίδιο ισχύει και για τη ROM του λειτουργικού συστήματος.

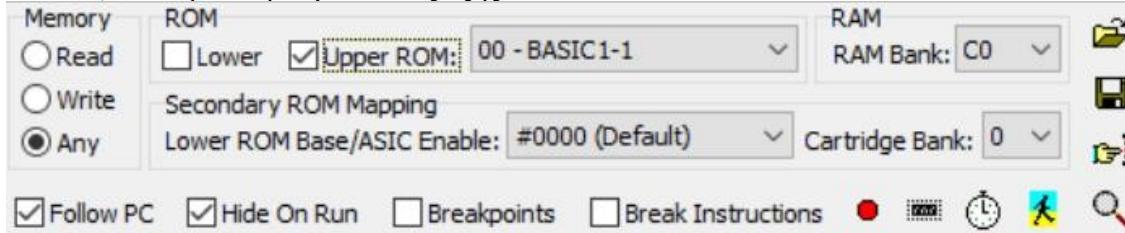
Από το παράθυρο εξομοίωσης πληκτρολογήστε
POKE &C000, &FF

Στη συνέχεια, πατήστε pause στον εξομοιωτή και θα δείτε μια οθόνη με διευθύνσεις και τιμές μνήμης. Αναζητήστε τη διεύθυνση &C000 και τα περιεχόμενά της. Θα πρέπει να είναι &FF. Στο κάτω μέρος θα δείτε κάτι σαν:



Σχ. 9 Winape με τη μνήμη RAM

Λοιπόν, αν τώρα το ρυθμίσετε ως εξής:



Σχ. 10 Winape με ROM

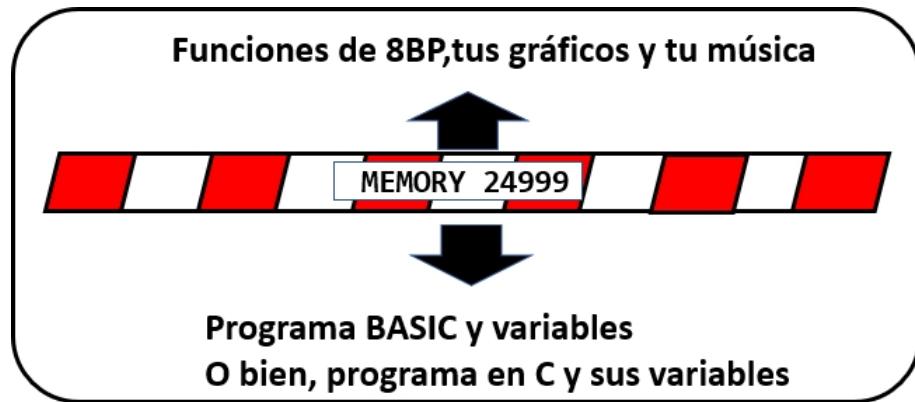
Θα δείτε τα περιεχόμενα της διεύθυνσης &C000 στην τράπεζα ROM που περιέχει τον διερμηνέα BASIC. Είναι πολύ ενδιαφέρον να μπορείτε να το κάνετε αυτό με το Winape.

2.4 Χάρτης μνήμης 8BP και επιλογές συναρμολόγησης

Δεδομένου ότι το Amstrad BASIC ξεκινά καταναλώνοντας τις χαμηλότερες διευθύνσεις, προκειμένου να "συνυπάρξουν", η βιβλιοθήκη 8BP φορτώνεται στην υψηλότερη διαθέσιμη περιοχή μνήμης. Είναι σημαντικό να καταλάβετε πώς λειτουργεί η BASIC για να χρησιμοποιήσετε τη βιβλιοθήκη, καθώς θα πρέπει να χρησιμοποιήσετε μια εντολή "MEMORY" της BASIC.

<pre> 10 a=5 20 k=Pa 30 PRINT k Ready run 411 Ready </pre>	<p>Το κείμενο ενός προγράμματος γραμμένου σε BASIC αποθηκεύεται ξεκινώντας από τη διεύθυνση &40 (σε δεκαδικό είναι 64, μια πολύ "χαμηλή" διεύθυνση) και οι μεταβλητές που δημιουργεί το πρόγραμμά σας αποθηκεύονται σε θέσεις αμέσως μετά τη μνήμη που καταλαμβάνει η λίστα. Το ακόλουθο παράδειγμα δείχνει πώς αποθηκεύεται η μεταβλητή "A", η οποία τυχαίνει να είναι η διεύθυνση 441.</p>
--	--

Η λίστα BASIC και οι μεταβλητές μπορεί να μεγαλώσουν πολύ, αν το πρόγραμμά σας είναι πολύ μεγάλο, και μπορεί να εισβάλουν στην περιοχή μνήμης όπου αποθηκεύονται οι ρουτίνες 8BP. Για να το αποφύγετε αυτό, πρέπει να εκτελέσετε μια εντολή **MEMORY** που προστατεύει την περιοχή μνήμης όπου αποθηκεύονται η 8BP, τα γραφικά σας και η μουσική σας. Η εντολή MEMORY είναι σαν ένα "φράγμα" που εμποδίζει τη BASIC και τις μεταβλητές της να καταλάβουν τη μνήμη πέρα από το όριο που της λέμε. Εάν το κάνει, θα δώσει ένα σφάλμα "MEMORY FULL" και θα σταματήσει να λειτουργεί.



Η βιβλιοθήκη της 8BP σας αφήνει ελεύθερα 24 έως 25 KB για την καταχώριση της BASIC ή το πρόγραμμα που μεταγλωττίσατε σε C (στην 8BP μπορείτε επίσης να προγραμματίσετε σε C). Από την έκδοση V42 της 8BP είναι δυνατόν να επιλέξετε διάφορες επιλογές συναρμολόγησης ανάλογα με τον τύπο του παιχνιδιού που πρόκειται να αναπτύξετε. Η ιδέα είναι απλή: αν πρόκειται να φτιάξετε ένα παιχνίδι με κύλιση χρειάζεστε τις εντολές που αφορούν την κύλιση, αλλά δεν χρειάζεστε τις εντολές που αφορούν τη σχεδίαση λαβυρίνθων (LAYOUT στο 8BP) ή την ανίχνευση συγκρούσεων με τους τοίχους του λαβυρίνθου. Δηλαδή, από την V42 και μετά, ορισμένες λειτουργίες της 8BP δεν θα χρησιμοποιούν μνήμη αν δεν χρειάζονται. Έτσι, η 8BP μπορεί να σας αφήσει περισσότερη ελεύθερη μνήμη για την καταχώριση της BASIC ή το πρόγραμμα C. Θα πρέπει να επιλέξετε μια επιλογή συναρμολόγησης ανάλογα με τον τύπο του παιχνιδιού που θέλετε να προγραμματίσετε. Αυτός ο πίνακας συνοψίζει τις διαθέσιμες επιλογές. Σύντομα θα καταλάβετε την εντολή SAVE που εμφανίζεται σε κάθε περιγραφή. Αυτός ο πίνακας είναι επίσης διαθέσιμος σε ένα παράρτημα.

Επιλογή	Περιγραφή της επιλογής	Παράδειγμα τυπικού παιχνιδιού
0	<p>Μπορείτε να κάνετε οποιοδήποτε παιχνίδι Όλες οι διαθέσιμες εντολές Πρέπει να χρησιμοποιήσετε το MEMORY 23499</p> <p>Για να αποθηκεύσετε τη βιβλιοθήκη + γραφικά + μουσική: SAVE "8BP0.bin",b,23500,19119</p>	<p>οποιοσδήποτε</p> 
1	<p>παιχνίδια λαβύρινθου ή διέλευσης οθόνης Πρέπει να χρησιμοποιήσετε τη μνήμη 24999</p> <p>Δεν διατίθεται σε αυτή τη λειτουργία:</p> <p> MAP2SP, UMAP, 3D</p> <p>Για να αποθηκεύσετε τη βιβλιοθήκη + γραφικά + μουσική: ΑΠΟΘΗΚΕΥΣΗ "8BP1.bin",b,25000,17619</p>	

	<p>Για παιχνίδια κύλισης Πρέπει να χρησιμοποιήσετε το MEMORY 24799 Δεν είναι διαθέσιμο σε αυτή τη λειτουργία:</p> <p> LAYOUT, COLAY, 3D</p> <p>Για να αποθηκεύσετε τη βιβλιοθήκη + γραφικά + μουσική:</p> <p>ΑΠΟΘΗΚΕΥΣΗ "8BP2.bin", b,24800,17819</p>	
	<p>Για παιχνίδια με ψευδο 3D Πρέπει να χρησιμοποιήσετε το MEMORY 23999 Δεν είναι διαθέσιμο σε αυτή τη λειτουργία:</p> <p> LAYOUT, COLAY</p> <p>Για να αποθηκεύσετε τη βιβλιοθήκη + γραφικά + μουσική:</p> <p>ΑΠΟΘΗΚΕΥΣΗ "8BP3.bin", b,24000,18619</p>	

Όπως μπορείτε να δείτε, η επιλογή που σας αφήνει τη μεγαλύτερη ελεύθερη μνήμη για το πρόγραμμά σας είναι η επιλογή 1, καθώς σας αφήνει 25KB ελεύθερα για το παιχνίδι σας. Η επιλογή 3 σας αφήνει 24KB και η επιλογή μηδέν σας αφήνει 23,5KB. Δεν σας συνιστώ να χρησιμοποιήσετε την επιλογή μηδέν, εκτός αν την χρειάζεστε πραγματικά. Είναι προτιμότερο να επιλέξετε τη σωστή επιλογή για το παιχνίδι σας και θα έχετε περισσότερη μνήμη διαθέσιμη.

Αυτά τα 24-25 KB που είναι ελεύθερα είναι για την καταχώρησή σας, καθώς η μουσική και τα γραφικά του παιχνιδιού σας αποθηκεύονται σε μια άλλη "υψηλότερη" περιοχή. Για παράδειγμα, με την επιλογή 1 έχετε:

- 25 KB δωρεάν για την καταχώρισή σας (BASIC ή C) και τις μεταβλητές σας
- 1.5 KB δωρεάν για τη μουσική σας
- 8.5 KB δωρεάν για τα γραφικά σας

ΣΥΝΟΛΟ: 35 KB δωρεάν για το παιχνίδι σας

Η επιλογή συναρμολόγησης που επιλέγετε ορίζεται σε ένα αρχείο 8BP που ονομάζεται "make_all_mygame.asm". Αυτό το αρχείο έχει μια γραμμή που **πρέπει να επεξεργαστείτε για να επιλέξετε την επιλογή που προτιμάτε**.

```

; Makefile para los videojuegos que usan 8bits de poder
; si alteras solo una parte solo tienes que ensamblar el make correspondiente
; por ejemplo puedes ensamblar el make_graficos si cambias dibujos

; DESDE LA V42 EXISTEN "OPCIONES" DE ENSAMBLAJE
; -----
; ASSEMBLING_OPTION = 0 --> todos los comandos disponibles.

; ASSEMBLING_OPTION = 1 --> para juegos de laberintos. MEMORY 25000
;                               disponibles los comandos |LAYOUT, |COLAY
;

; ASSEMBLING_OPTION = 2 --> para juegos con scroll, MEMORY 24800
;                               disponibles los comandos |MAP2SP, |UMA
;

; ASSEMBLING_OPTION = 3 --> para juegos pseudo 3D , MEMORY 24000
;                               disponible comando |3D
;

; ASSEMBLING_OPTION = 4 --> uso futuro

let ASSEMBLING_OPTION = 1
;-----CODIGO-----
;incluye la libreria 8bp y el playerWYZ de musica
read "make_codigo_mygame.asm"

;-----MUSICA-----
; incluye las canciones.
read "make_musica_mygame.asm"

; ----- GRAFICOS -----
; esta parte incluye imagenes y secuencias de animacion
; y la tabla de sprites inicializada con dichas imagenes y secuencias
read "make_graficos_mygame.asm"

```

Πριν από τη φόρτωση της βιβλιοθήκης, πρέπει να εκτελέσετε την εντολή **MEMORY** με το όριο που σχετίζεται με τον τρόπο συναρμολόγησης που έχετε επιλέξει για τον τύπο του παιχνιδιού σας. Από την έκδοση V42 υπάρχουν 24KB, 24,8 KB ή 25KB ελεύθερα, ανάλογα με την επιλογή συναρμολόγησης που έχετε επιλέξει.

Ανεξάρτητα από την επιλογή συναρμολόγησης που θα επιλέξετε, οι διευθύνσεις μνήμης

στις οποίες βρίσκονται όλες οι εντολές είναι ακριβώς οι ίδιες (Παράρτημα XI). Για το

Για παράδειγμα, η εντολή LAYOUT μπορεί να κληθεί εάν επιλέξετε την επιλογή συναρμολόγησης 1 ή 2, αλλά εάν χρησιμοποιήσετε την επιλογή συναρμολόγησης 2 , η κλήση της εντολής LAYOUT δεν θα κάνει απολύτως τίποτα, όπως και η εντολή MAP2SP εάν χρησιμοποιήσετε την επιλογή συναρμολόγησης 1.

```

ΧΑΡΤΗΣ ΜΝΗΜΗΣ AMSTRAD CPC464 της 8BP

; &FFFF +-----  

; | οθόνη + 8 κρυφά τμήματα των 48bytes το καθένα  

; &C000 +-----  

; | σύστημα (επαναπροσδιορίσιμα σύμβολα, δείκτης  

στοίβας, κ.λπ.)-----  

; 42619 +-----  

; | -Τράπεζα 40 αστέρων (από 42540 έως 42619 = 80bytes)  

; 42540 +-----  

; | Χάρτης διάταξης χαρακτήρων (25x20 =500 bytes)  

; | και παγκόσμιος χάρτης (έως 82 στοιχεία χωράνε σε 500 bytes)  

; | Και τα δύο αποθηκεύονται στην ίδια περιοχή μνήμης.  

; | -γιατί είτε-χρησιμοποιείτε το ένα είτε χρησιμοποιείτε το άλλο.  

; 42040 +-----sprites (σχεδόν 8,5KB για σχέδια).  

; | έχετε 8440 bytes αν δεν υπάρχουν ακολουθίες και  

; | διαδρομές)  

; +-----Οι εικόνες του αλφαριθμητικού αποθηκεύονται επίσης εδώ.  

; | Ορισμοί διαδρομών (μεταβλητού μήκους ο καθένας)  

; +-----  

; | ακολουθίες κινούμενων σχεδίων των 8 καρέ (16 bytes το  

; | καθένα)  

; 33600 +-----κειτ-θητέδες ακολουθιών κινούμενων σχεδίων (μακρο-  

; | φιλαρχοθέτη)  

; | (1500 Bytes για μουσική που έχει επεξεργαστεί με το  

; 32100 +-----WYZtracker 2.0.1.0)  

; | Ρουτίνες 8BP (8100 bytes ή 7100 bytes)  

; | Εδώ είναι όλες οι ρουτίνες και ο πίνακας sprite  

; | περιλαμβάνει music player "wyz" 2.0.1.0  

; 25000 +-----  

; |  

; | Η ΛΙΣΤΑ BASIC ή C ΣΑΣ  

; | 24KB, 24,8 KB ή έως και 25KB ελεύθερα για BASIC ή C,  

; | ανάλογα με την επιλογή συναρμολόγησης που  

; | χρησιμοποιείτε για την 8BP  

; 0 +-----
```

Σχ. 11 Μνήμη με χρήση 8BP

Αν σας τελειώσει ο χώρος για τα γραφικά ή τη μουσική και χρειάζεστε περισσότερο, η 8BP σας επιτρέπει να τοποθετήσετε εικόνες και μουσική σε άλλες περιοχές (κάτω από τη διεύθυνση 24000) και θα λειτουργήσει μια χαρά.

3 Απαραίτητα εργαλεία

Winape: εξομοιωτής για το λειτουργικό σύστημα Windows με επεξεργαστή για να επεξεργαστείτε και να δοκιμάσετε το πρόγραμμα BASIC σας. Και επίσης για τη συναρμολόγηση γραφικών και μουσικής.

SPEDIT: ("Simple Sprite Editor") Εργαλείο BASIC για την επεξεργασία των γραφικών σας. Η έξοδος του spedit είναι κώδικας assembler που αποστέλλεται στον εκτυπωτή Amstrad CPC. Εκτελώντας το εργαλείο μέσα στο Winape, ο εκτυπωτής ανακατευθύνεται σε ένα αρχείο κειμένου, έτσι ώστε τα γραφικά σας να αποθηκεύονται σε ένα αρχείο txt. Αυτό το εργαλείο δημιουργήθηκε για να συμπληρώσει τη βιβλιοθήκη 8BP και τα γραφικά όλων των παιχνιδιών που έχω δημιουργήσει έχουν γίνει με το SPEDIT.

Wyztracker: για τη σύνθεση μουσικής, κάτω από τα παράθυρα. Το πρόγραμμα που μπορεί να αναπαράγει τις μελωδίες που συνθέτει το Wyztracker είναι το Wyzplayer, το οποίο είναι ενσωματωμένο στο 8BP. Αφού συνθέσετε τη μουσική, μπορείτε να την αναπαράγετε με μια απλή εντολή |MUSIC

Βιβλιοθήκη 8BP: εγκαταστήστε νέες εντολές προσβάσιμες από τη BASIC για το πρόγραμμά σας. Όπως θα δείτε, αυτή θα είναι η "καρδιά" που θα κινεί τα μηχανήματα που θα φτιάξετε.

CPCDiskXP : σας επιτρέπει να εγγράψετε μια δισκέτα 3,5", την οποία μπορείτε στη συνέχεια να τοποθετήσετε στο CPC6128 σας, αν έχετε καλώδιο για τη σύνδεση μονάδας δισκέτας. Αν θέλετε να δημιουργήσετε μια κασέτα ήχου CPC464, αυτό το εργαλείο δεν είναι απαραίτητο.

2CDT: απαραίτητο εργαλείο για τη δημιουργία αρχείων .cdt. Συνήθως εξάγω τα αρχεία από ένα .dsk σε δίσκο των Windows χρησιμοποιώντας το CPCDiskXP και στη συνέχεια χρησιμοποιώ το 2cdt για να δημιουργήσω το αρχείο cdt.

Tape2wav: εργαλείο που σας επιτρέπει να δημιουργήσετε αρχεία .wav από αρχεία .cdt

ΠΡΟΑΙΡΕΤΙΚΑ:

ConvImgCPC: επεξεργαστής εικόνων για τα παιχνίδια σας. Μετατρέπει επίσης από BMP. Προγραμματισμένος από τον Ludovic Deplanque ("DEMONIAK")

RGAS: (Retro Game Asset Studio) πανίσχυρος επεξεργαστής sprite, που εξελίχθηκε από το εργαλείο AMSprite και δημιουργήθηκε από τον Lachlan Keown. Αυτός ο επεξεργαστής sprite είναι συμβατός με 8BP και τρέχει υπό Windows. Όταν ξεπεράσετε το Spedit, αυτό μπορεί να είναι η καλύτερη επιλογή.

ΔΕΝ ΣΥΝΙΣΤΑΤΑΙ:

Fabacom: εκτελέσιμο πρόγραμμα μεταγλώττισης μέσα στο AMSTRAD CPC 6128 ή από τον εξομοιωτή Winape για να μεταγλωττίσετε το πρόγραμμα BASIC σας και να το

κάνετε να εκτελεστεί γρηγορότερα. Είναι συμβατό με τις κλήσεις εντολών της βιβλιοθήκης 8BP. Ωστόσο, δεν συνιστάται για διάφορους λόγους:

- Το πρόγραμμά σας θα καταλαμβάνει πολύ περισσότερο χώρο επειδή το fabacom χρειάζεται επιπλέον 10KB για τις βιβλιοθήκες του, και επίσης, όταν μεταγλωττίσει το πρόγραμμά σας, εξακολουθεί να καταλαμβάνει τον ίδιο χώρο.

έτσι ώστε ένα πρόγραμμα BASIC των 10KB να μετατραπεί σε πρόγραμμα BASIC των 20KB.

- Υπάρχουν ορισμένα καταγεγραμμένα προβλήματα ασυμβατότητας αυτού του μεταγλωττιστή με ορισμένες εντολές BASIC.
- Επιπλέον, όπως θα δείτε σε αυτό το βιβλίο, μπορείτε να επιτύχετε πολύ υψηλή ταχύτητα χωρίς μεταγλώττιση.

Μεταγλωττιστής CPC BASIC: εκτελέσιμος μεταγλωττιστής για windows. Είναι συμβατός με τις κλήσεις εντολών της βιβλιοθήκης 8BP. Σε αντίθεση με το fabacom, το μεταγλωττισμένο πρόγραμμα καταλαμβάνει μόνο περίπου 5KB επιπλέον, ωστόσο, διατηρεί 16KB εργασίας για να εκτελέσει, έτσι ώστε να μην αφήνει σχεδόν καθόλου χώρο για το πρόγραμμά σας, αφού "κλέβει" συνολικά 20 KB. Και δεν είναι 100% συμβατό με την locomotive BASIC.

Η αύξηση της ταχύτητας τόσο με το fabacom όσο και με τον μεταγλωττιστή CPC Basic μπορεί να φτάσει το 50%, ανάλογα με το παιχνίδι. Δηλαδή, ένα παιχνίδι που τρέχει με 20 FPS θα τρέχει με 30 FPS. Αυτό δεν είναι κακό, αλλά σκεφτείτε ότι έχουμε περάσει από την ερμηνευμένη BASIC στον κώδικα μηχανής και κανονικά λέγεται ότι η ταχύτητα πρέπει να πολλαπλασιαστεί τουλάχιστον επί 100 (θα μιλούσαμε για αύξηση 10000%). Ωστόσο, έχουμε κερδίσει μόνο 50%. Ο λόγος για ένα τόσο "φτωχό" κέρδος είναι ότι οι εντολές 8BP κάνουν ήδη όλη τη δύσκολη δουλειά και στην πραγματικότητα ο μεταγλωττιστής μεταφράζει σε κώδικα μηχανής μόνο το λιγότερο βαρύ μέρος, τη λογική του παιχνιδιού.

Τέλος, θα πρέπει να γνωρίζετε ότι, αν θέλετε τα προγράμματά σας να τρέχουν πολύ πιο γρήγορα, από την έκδοση v40 του 8BP υπάρχει υποστήριξη της γλώσσας C, οπότε μπορείτε είτε να προγραμματίσετε ολόκληρο το παιχνίδι σας απευθείας σε C είτε να προγραμματίσετε σε BASIC και να μεταφράσετε μόνο τον "κύκλο του παιχνιδιού" σε C. Υπάρχει ένα κεφάλαιο σε αυτό το εγχειρίδιο αφιερωμένο αποκλειστικά σε αυτό το θέμα. Υπάρχει ένα κεφάλαιο σε αυτό το εγχειρίδιο αφιερωμένο αποκλειστικά σε αυτό το θέμα.

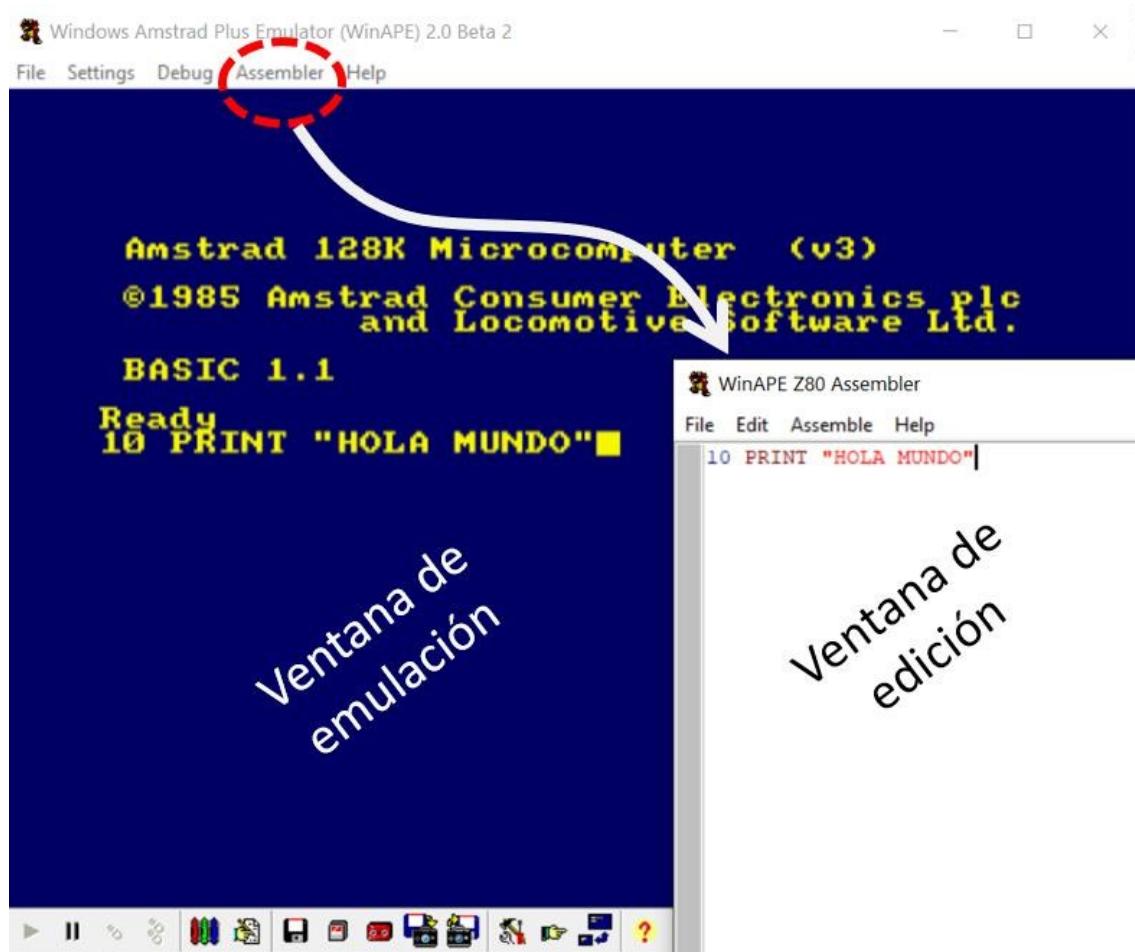
4 Πρώτα βήματα με την 8BP

4.1 Εγκαταστήστε το winape

Το πρώτο πράγμα που πρέπει να κάνετε είναι να εγκαταστήσετε την τελευταία έκδοση του **Winape**, το οποίο είναι ένας εξομοιωτής, επεξεργαστής και συναρμολογητής Amstrad. Μπορείτε να το κατεβάσετε από τη διεύθυνση www.winape.net.

4.2 Εξοικείωση με το winape: "hello world".

Μόλις εγκατασταθεί το Winape, εξοικειωθείτε με αυτό δοκιμάζοντας μερικά παιχνίδια Amstrad και προσπαθώντας να αλλάξετε τις ρυθμίσεις. Δοκιμάστε να ανοίξετε το ενσωματωμένο μενού του assembler και να επεξεργαστείτε ένα "hello world" στο παράθυρο του assembler. Στη συνέχεια, αντιγράψτε και επικολλήστε το κείμενο στο παράθυρο εξομοίωσης. Θα δείτε πώς επικολλάται χαρακτήρας προς χαρακτήρα



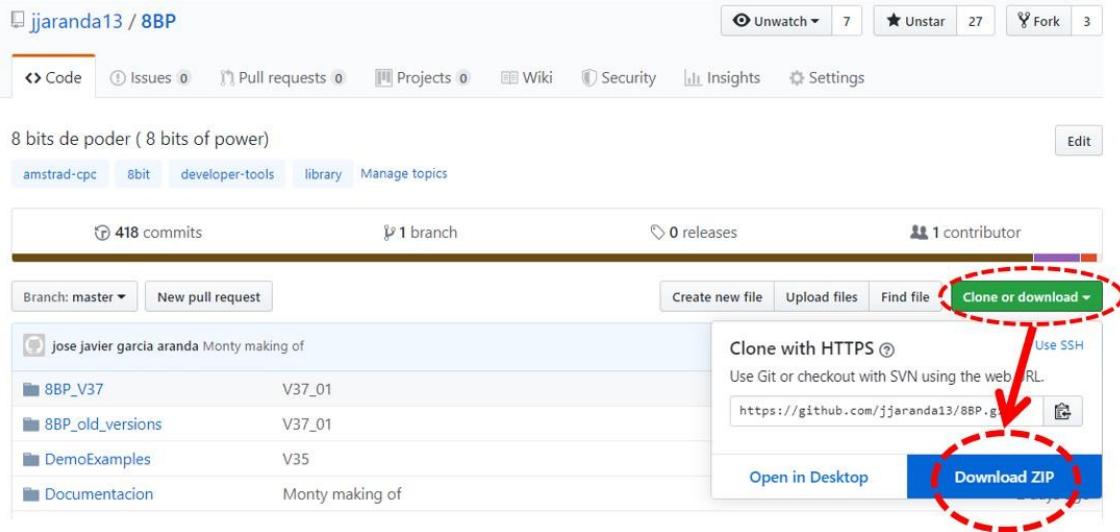
Αν φτιάξετε ένα πρόγραμμα μεγαλύτερης διάρκειας, για να το αντιγράψετε στο παράθυρο εξομοίωσης, είναι πολύ ενδιαφέρον να χρησιμοποιήσετε την επιλογή settings->high speed. Θα δείτε πόσο γρήγορα αντιγράφεται.

4.3 Κατεβάστε τη βιβλιοθήκη 8BP

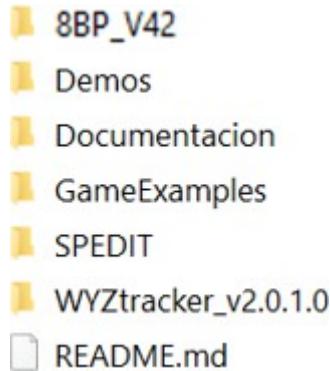
Φτάνουμε στο πιο ενδιαφέρον σημείο. Πηγαίνετε στη διεύθυνση

<https://github.com/jjaranda13/8BP> και κατεβάστε το zip
(<https://github.com/jjaranda13/8BP/archive/master.zip>) . Για να το κάνετε αυτό

μπορείτε απλά να κάνετε κλικ στο πράσινο κουμπί "κλωνοποίηση ή λήψη" και στη συνέχεια να επιλέξετε το αρχείο zip.



Μόλις το κατεβάσετε, αποσυμπίεστε το στον κατάλογο της επιλογής σας. Θα λάβετε το ακόλουθο αποτέλεσμα:



4.4 Εκτελέστε τα demo

Τώρα θα κάνουμε μια πρώτη επαφή με την 8BP παρακολουθώντας μερικά demo. Πηγαίνετε στον κατάλογο Demos. Σε αυτόν θα βρείτε μια σειρά από υποφακέλους: **ASM, BASIC, C, DSK, MUSIC, ASM, BASIC, C, DSK**.

Πηγαίνετε στο DSK. Εκεί θα δείτε ένα αρχείο .dsk με τα demo. Από το winape μεταβείτε στο μενού File.

>drive A-> insert Disc image και επιλέξτε το αρχείο demo

Μόλις επιλεγεί, από το παράθυρο εξόμοιωσης Amstrad, πληκτρολογήστε **CAT** για να δείτε τα αρχεία.

cat

Drive A: user 0

8BP0	.BIN	18K	DEMO15	.BAS	2K
8BP1	.BIN	18K	DEMO2	.BAS	2K
8BP2	.BIN	19K	DEMO3	.BAS	1K
CICLO	.BIN	4K	DEMO4	.BAS	2K
DEMO1	.BAS	2K	DEMO5	.BAS	2K
DEMO10	.BAS	2K	DEMO6	.BAS	1K
DEMO11	.BAS	1K	DEMO7	.BAS	2K
DEMO11	.BIN	1K	DEMO8	.BAS	1K
DEMO12	.BAS	3K	DEMO9	.BAS	1K
DEMO13	.BAS	2K	LOADER	.BAS	2K
DEMO14	.BAS	3K			

89K free

Ready

Κάθε αρχείο .BAS είναι ένα demo όπου μπορείτε να δείτε μερικά από τα χαρακτηριστικά της 8BP (δεν μπορείτε να δείτε όλα τα χαρακτηριστικά στα demo, αλλά υπάρχουν μερικά αντιπροσωπευτικά).

Εκτελέστε την εντολή **RUN "LOADER.BAS"**. Θα εμφανιστεί το ακόλουθο μενού:

```
***** ****
* * * * * *
* * * * * *
*** *** ***
* * * * *
* * * * *
*** *** *** v42
elige una demo
-----
1) juego sencillo con musica
2) juego sencillo con salto y disparo
3) test colision sprites
4) test layout con sobreescritura
5) test animacion tintas
6) test sobreescritura mode 1
7) test ordenamiento sprites
8) test mini alfabeto
9) test rutas de sprites
10) test scroll multidireccional
11) test pseudo-3D
12) test stars
13) test music y fx a la vez
14) test 8BP en C
15) test background images
?
```

Μπορείτε τώρα να επιλέξετε ένα demo και να το δοκιμάσετε. Απολαύστε το. Στο επόμενο και τελευταίο βήμα θα ξεκινήσουμε τη δημιουργία ενός παιχνιδιού.

4.5 Δημιουργία του πρώτου σας προγράμματος με την 8BP

Δοκιμάσαμε ένα αρχείο .dsk που περιέχει πολλά demo, με γραφικά και μουσική. Ο κατάλογος Demos/ASM και ο κατάλογος Demos/MUSIC περιέχουν τα γραφικά και τη μουσική των demo που έχετε δοκιμάσει. Ωστόσο, αν θέλετε να φτιάξετε το δικό σας παιχνίδι ή demo, είναι προτιμότερο να ξεκινήσετε με "καθαρά" αρχεία χωρίς όλα τα γραφικά που απαιτούνται από τα demo που έχετε δοκιμάσει.

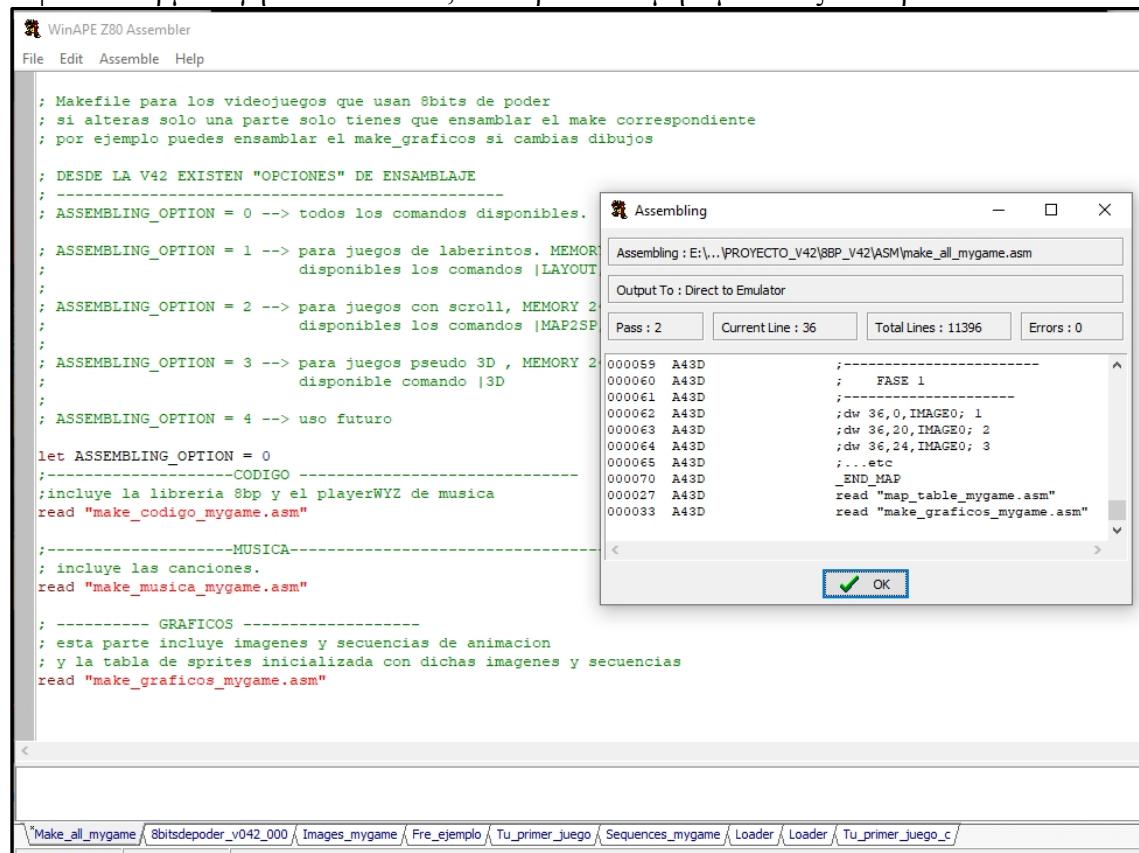
Πηγαίνουμε στον ριζικό κατάλογο. Εκεί θα βρείτε έναν φάκελο με την ονομασία "**8BP_V42**". Σας συνιστώ να δημιουργήσετε ένα αντίγραφο αυτού του φακέλου και να τον μετονομάσετε σε "**my_game**". Με αυτόν τον τρόπο θα διατηρήσετε τον αρχικό φάκελο "**8BP_V42**", ακόμα και αν αρχίσετε να αλλάζετε πράγματα.



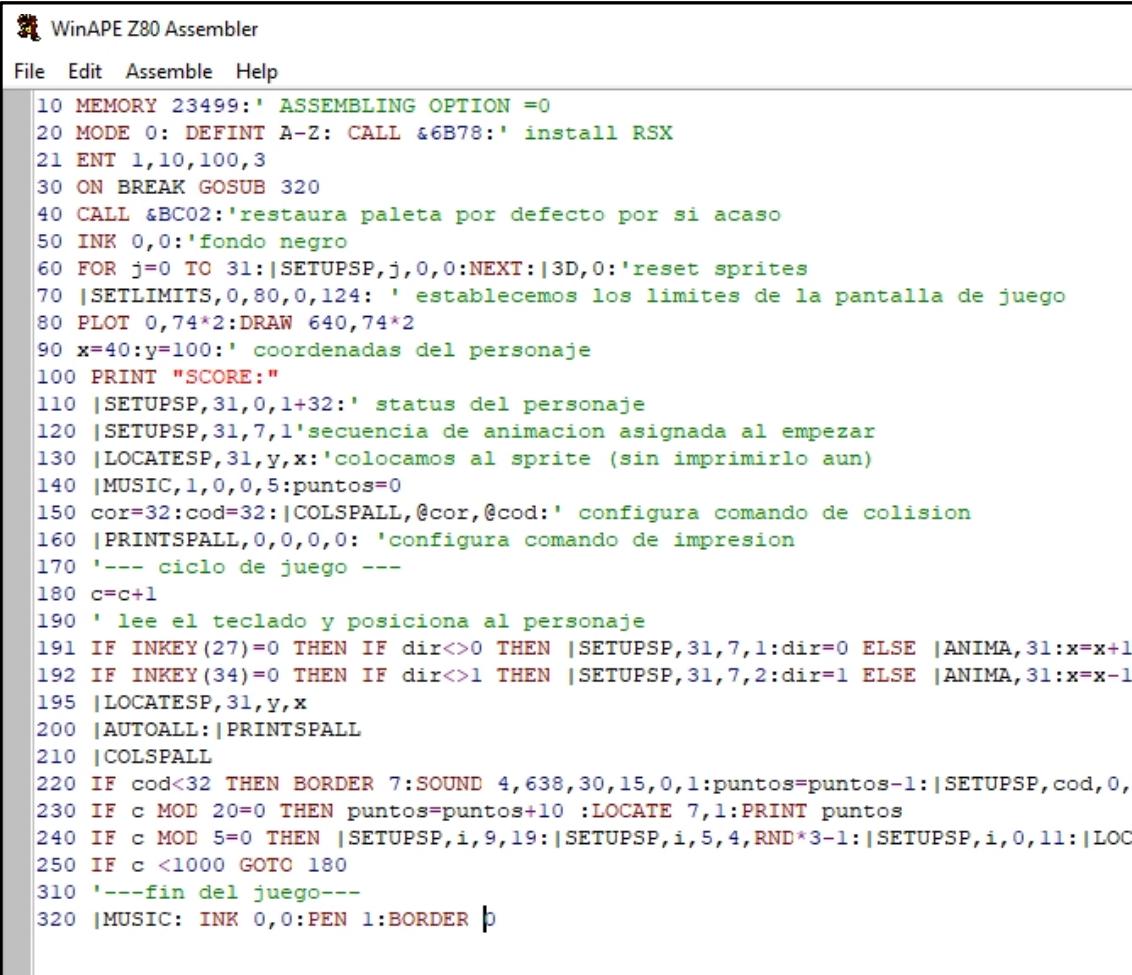
Μέσα στο φάκελο "**8BP_V42**" θα βρείτε τους φακέλους ASM, BASIC, DSK, MUSIC, TAPE και output_spedit.

Από το παράθυρο του winape assembler ανοίξτε το αρχείο "ASM/make_all_mygame.asm" και εκτελέστε το (απλά στο μενού του winape z80 assembler επιλέξτε "Assemble" ή πατήστε Ctrl+F9). Αυτό θα ξεκινήσει τη συναρμολόγηση (αντιγραφή στη μνήμη) της βιβλιοθήκης και των γραφικών στη μνήμη του Amstrad. Σε αυτή την περίπτωση θα συναρμολογήσουμε πολύ λίγα γραφικά, μόνο τα απαραίτητα για ένα μικρό παιχνίδι. Τα γραφικά βρίσκονται στο αρχείο "images_mygame.asm".

Αφού συναρμολογήσετε τα πάντα, θα λάβετε ένα μήνυμα όπως το παρακάτω:



Πατήστε "ok" και από το παράθυρο του assembler θα ανοίξουμε ένα άλλο αρχείο. Σε αυτή την περίπτωση θα ανοίξουμε ένα αρχείο BASIC, συγκεκριμένα το "**your_first_game.bas**" το οποίο βρίσκεται στο φάκελο BASIC. Αφού το ανοίξετε, θα δείτε στην οθόνη την ακόλουθη λίστα, η οποία περιέχει 32 γραμμές:



```

WinAPE Z80 Assembler
File Edit Assemble Help
10 MEMORY 23499:' ASSEMBLING OPTION =0
20 MODE 0: DEFINT A-Z: CALL &6B78:' install RSX
21 ENT 1,10,100,3
30 ON BREAK GOSUB 320
40 CALL &BC02:'restaura paleta por defecto por si acaso
50 INK 0,0:'fondo negro
60 FOR j=0 TO 31:|SETUPSP,j,0,0:NEXT:|3D,0:'reset sprites
70 |SETLIMITS,0,80,0,124: ' establecemos los limites de la pantalla de juego
80 PLOT 0,74*2:DRAW 640,74*2
90 x=40:y=100:' coordenadas del personaje
100 PRINT "SCORE:"
110 |SETUPSP,31,0,1+32:' status del personaje
120 |SETUPSP,31,7,1'secuencia de animacion asignada al empezar
130 |LOCATESP,31,y,x:'colocamos al sprite (sin imprimirlo aun)
140 |MUSIC,1,0,0,5:puntos=0
150 cor=32:cod=32:|COLSPALL,@cor,@cod:' configura comando de colision
160 |PRINTSPALL,0,0,0,0: 'configura comando de impresion
170 '--- ciclo de juego ---
180 c=c+1
190 ' lee el teclado y posiciona al personaje
191 IF INKEY(27)=0 THEN IF dir<>0 THEN |SETUPSP,31,7,1:dir=0 ELSE |ANIMA,31:x=x+1
192 IF INKEY(34)=0 THEN IF dir<>1 THEN |SETUPSP,31,7,2:dir=1 ELSE |ANIMA,31:x=x-1
195 |LOCATESP,31,y,x
200 |AUTOALL:|PRINTSPALL
210 |COLSPALL
220 IF cod<32 THEN BORDER 7:SOUND 4,638,30,15,0,1:puntos=puntos-1:|SETUPSP,cod,0,
230 IF c MOD 20=0 THEN puntos=puntos+10 :LOCATE 7,1:PRINT puntos
240 IF c MOD 5=0 THEN |SETUPSP,i,9,19:|SETUPSP,i,5,4,RND*3-1:|SETUPSP,i,0,11:|LOC
250 IF c <1000 GOTO 180
310 '---fin del juego---
320 |MUSIC: INK 0,0:PEN 1:BORDER |

```

Επιλέξτε τα πάντα και αντιγράψτε τα. Στη συνέχεια, μεταβείτε στο παράθυρο εξόμοιώσης CPC και επικολλήστε το χρησιμοποιώντας το μενού **ΑΡΧΕΙΟ->Επικόλληση**.

Καθώς η λίστα είναι λίγο μεγαλύτερη από το "hello world", χρησιμοποιήστε το μενού winape και επιλέξτε **settings->high speed** για να την αντιγράψετε και μετά επιστρέψτε στην "κανονική ταχύτητα". Καθώς η βιβλιοθήκη και τα γραφικά είναι ήδη συναρμολογημένα, μπορείτε να RUN και το παιχνίδι θα τρέξει. Πρέπει να αποφύγετε τις μπάλες που πέφτουν από τον ουρανό για να μην πεθάνετε, κινούμενοι αριστερά και δεξιά.



Μπορείτε να δοκιμάσετε να τροποποιήσετε το πρόγραμμα και να δείτε τα αποτελέσματά του. Σιγά-σιγά θα μάθετε το 8BP και θα μπορείτε να κάνετε ενδιαφέρουσες τροποποιήσεις, όπως να αλλάξετε τη συχνότητα με την οποία βγαίνουν οι εχθρικές μπάλες ή την ταχύτητά τους ή να αντικαταστήσετε τον στρατιώτη με ένα διαστημόπλοιο και τις μπάλες με εχθρικά πλοία με διαφορετικές τροχιές.

Αν θέλετε να δοκιμάσετε πόσο γρήγορα λειτουργεί σε γλώσσα C, απλά φορτώστε το .dsk και εκτελέστε το "loader.bas", θα φορτώσει μια έκδοση του παιχνιδιού που σας επιτρέπει να επιλέξετε μεταξύ της έκδοσης BASIC και της έκδοσης C, ώστε να μπορείτε να συγκρίνετε. Υπάρχει ένα κεφάλαιο σε αυτό το βιβλίο αφιερωμένο στον προγραμματισμό σε C χρησιμοποιώντας την 8BP και μια μίνι BASIC, ώστε να μπορείτε να προγραμματίζετε σε C όπως ακριβώς θα κάνατε και σε BASIC. Αν δεν έχετε μεγάλη εμπειρία στον προγραμματισμό σε C, σας συνιστώ να προχωρήσετε αργά και να προγραμματίσετε σε BASIC, τα αποτελέσματα θα είναι γρήγορα και επαγγελματικά.

4.6 Δημιουργήστε το .dsk με το σετ 8BP

Τέλος, θα δημιουργήσουμε έναν δίσκο με το παιχνίδι σας. Για να το κάνετε αυτό, αφού εκτελέστε το παιχνίδι, πρέπει να ακολουθήσετε τα εξής βήματα

- Δημιουργήστε έναν νέο δίσκο μέσω του winape: ΑΡΧΕΙΟ-> μονάδα δίσκου A-> νέος δίσκος Blanc
- Μορφοποιήστε το: ΑΡΧΕΙΟ->Δίσκος A->Μορφοποίηση εικόνας δίσκου
- Αφού δημιουργήσετε το αρχείο dsk, από το παράθυρο εξόμοιωσης, εκτελέστε τις ακόλουθες εντολές:

ΑΠΟΘΗΚΕΥΣΗ "8BP0.bin", b, 23499,19119

ΑΠΟΘΗΚΕΥΣΗ "juego.bas"

Η πρώτη εντολή αποθηκεύει τη βιβλιοθήκη 8BP, τα γραφικά και τη μουσική στο δίσκο. Σε αυτή την περίπτωση χρησιμοποιούμε την επιλογή συναρμολόγησης 0 (βλέπε επιλογές στο προηγούμενο κεφάλαιο), αλλά θα μπορούσατε να χρησιμοποιήσετε οποιαδήποτε επιλογή για αυτό το παράδειγμα απολύτως καλά. Έχω ονομάσει το αρχείο **"8BP0"** για να αντικατοπτρίζει κατά κάποιο τρόπο ότι έχει χρησιμοποιηθεί η επιλογή συναρμολόγησης 0, αλλά το όνομα αυτού του δυαδικού αρχείου μπορεί να είναι οτιδήποτε.

Σχεδόν τελειώσαμε. Τώρα πρέπει να επιλέξετε έναν άλλο δίσκο από το μενού winape ή να βγείτε από το winape, ώστε το .dsk που δημιουργήσατε να γίνει πραγματικότητα στο σύστημα αρχείων των windows.

Βγείτε από το winape και ανοίξτε το ξανά.

Επιλέξτε το δίσκο που δημιουργήσατε και εκτελέστε:

MNHMH 23499

ΦΟΡΤΙΟ "8BP0.BIN"

**RUN "game.bas" RUN "game.bas" RUN "game.bas" RUN "game.bas" RUN
"game.bas" RUN**

Αλληλούια!

5 Βήματα για να φτιάξετε ένα παιχνίδι

5.1 Δομή καταλόγου του έργου σας

Όταν προγραμματίζετε το παιχνίδι σας, συνιστάται να διαρθρώνετε τα διάφορα αρχεία σε 7 φακέλους, ανάλογα με τον τύπο του αρχείου που είναι.

Είναι απολύτως δυνατό να τα βάλετε όλα στον ίδιο κατάλογο και να εργαστείτε χωρίς φακέλους, ωλλά είναι πιο "καθαρό" να το κάνετε με τον τρόπο που θα σας παρουσιάσω

- ASM
- BASIC
- C
- dsk
- MUSIC
- output_spedit

παρακάτω.

Σχ. 12 Δομή καταλόγου

- **ASM:** εδώ θα τοποθετήσετε αρχεία κειμένου γραμμένα σε assembler (.asm), όπως η ίδια η βιβλιοθήκη 8BP, τα sprites που δημιουργούνται με τον επεξεργαστή SPEDIT και μερικά βοηθητικά αρχεία.
- **BASIC:** εδώ θα βάλετε το παιχνίδι σας και βοηθητικά προγράμματα όπως το SPEDIT και το Loader.
- **C:** αυτός ο φάκελος είναι για "προχωρημένους" χρήστες που θέλουν να προγραμματίσουν σε C ολόκληρο το παιχνίδι ή τουλάχιστον τον κύκλο του παιχνιδιού. Είναι απαραίτητος μόνο αν πρόκειται να προγραμματίσετε κάποιο μέρος του παιχνιδιού σας σε γλώσσα C.
- **Dsk:** εδώ θα τοποθετήσετε το αρχείο .dsk έτοιμο να τρέξει σε ένα Amstrad CPC. Μέσα σε αυτό θα πρέπει να τοποθετήσετε 5 αρχεία για τα οποία θα μιλήσουμε στην επόμενη ενότητα
- **Μουσική:** με το μουσικό sequencer του WYZtracker, μπορείτε να δημιουργήσετε τα τραγούδια σας και να τα αποθηκεύσετε σε μορφή .wyz σε αυτόν τον κατάλογο. Μόλις τα "εξάγετε", θα δημιουργηθεί ένα αρχείο .asm που θα πρέπει να αποθηκεύσετε στο φάκελο ASM και ένα δυαδικό αρχείο που θα αποθηκεύσετε επίσης στο φάκελο ASM (μπορείτε επίσης να τα αφήσετε σε αυτόν το φάκελο, αρκεί να τα αναφέρετε σωστά στο αρχείο make_musica.asm στο φάκελο ASM).
- **Output_spedit:** σε αυτόν το φάκελο μπορείτε να αποθηκεύσετε το αρχείο κειμένου που δημιουργείται από το spedit. Το SPEDIT στέλνει τα sprites στον εκτυπωτή σε μορφή assembler και ο εξομοιωτής winape μπορεί να συλλέξει την έξοδο από τον εκτυπωτή Amstrad σε ένα αρχείο. Εδώ θα το τοποθετήσουμε
- **Ταινία:** εδώ μπορείτε να αποθηκεύσετε το αρχείο .wav, αν θέλετε να δημιουργήσετε μια κασέτα για να τη φορτώσετε στο Amstrad CPC464, ή το αρχείο .cdt.

5.2 Το παιχνίδι σας σε μόλις 3 αρχεία

Για να δημιουργήσετε το παιχνίδι σας θα χρειαστείτε ένα δυαδικό αρχείο και δύο αρχεία BASIC. Μπορείτε να δημιουργήσετε πολλά ανεξάρτητα δυαδικά αρχεία (ένα με τη βιβλιοθήκη 8BP, ένα με τα γραφικά, ένα με τη μουσική...) αλλά καθώς αυτές οι περιοχές μνήμης είναι συνεχόμενες, είναι προτιμότερο να δημιουργήσετε ένα ενιαίο δυαδικό αρχείο.

Το δυαδικό αρχείο περιέχει τη βιβλιοθήκη, τη μουσική, τα γραφικά, την περιοχή μνήμης ή τη διάταξη του παγκόσμιου χάρτη και προαιρετικά την τράπεζα αστεριών. Η ιδέα είναι να αποθηκεύσετε όλα τα δυαδικά συστατικά μαζί σε ένα αρχείο, όπως αυτό (ανάλογα με την επιλογή συναρμολόγησης θα χρησιμοποιήσετε τη μία ή την άλλη από αυτές τις εντολές). Το όνομα του αρχείου τελειώνει με έναν αριθμό που υποδεικνύει την επιλογή συναρμολόγησης, αλλά μπορεί να ονομαστεί όπως θέλετε.

```
SAVE "yourgame0.bin",b,2350 19119
      0,
SAVE "yourgame1.bin",b,2500 17619
      0,
SAVE "yourgame2.bin",b,2480 17819
      0,
SAVE "yourgame3.bin",b,2400 18619
      0,
```

Το γεωγραφικό μήκος είναι η τελική διεύθυνση μείον την αρχική διεύθυνση. Η τελική διεύθυνση, συμπεριλαμβανομένης της μουσικής και των γραφικών, του χάρτη και της τράπεζας αστεριών, είναι 42619, οπότε το μήκος μπορεί να υπολογιστεί αφαιρώντας την αρχική διεύθυνση από το 42620. Για παράδειγμα, στην επιλογή συναρμολόγησης 1, έχουμε $42619 - 25000 = 17619$, ακριβώς το μήκος που εμφανίζεται σε αυτή την εντολή.

Το δεύτερο αρχείο είναι το βασικό σας σύνολο
ΑΠΟΘΗΚΕΥΣΗ "tujuego.bas"

Και το τρίτο αρχείο είναι ο φορτωτής ("loader.bas"), ο οποίος θα είναι απλά:

```
10 MNHMH 23499
15 LOAD "!pant.scr",&c000: 'μόνο αν το παιχνίδι σας έχει οθόνη
φόρτωσης
20 LOAD " yourgame0.bin"
50 RUN " !yourgame.bas"
```

Έχω τοποθετήσει ένα αρχικό φορτίο οθόνης στη διεύθυνση μνήμης αρχικής οθόνης (&C000). Στο τέλος αυτού του εγχειριδίου θα βρείτε έναν από τους πολλούς τρόπους για να κάνετε ένα φορτίο οθόνης. Είναι προαιρετικός. Μπορείτε να φτιάξετε ένα παιχνίδι με ή χωρίς οθόνη φόρτωσης.

Αυτή η μέθοδος των 3 αρχείων είναι χρήσιμη ειδικά αν καταλαμβάνετε σχεδόν όλη τη διαθέσιμη μνήμη για γραφικά. Στα παιχνίδια με κασέτες η φόρτωση των 18KB μπορεί να πάρει αρκετή ώρα και αν δεν χρησιμοποιείτε τα 8,5KB γραφικών, ίσως είναι καλύτερα να φορτώσετε τα διάφορα δυαδικά αρχεία ξεχωριστά για να εξοικονομήσετε χρόνο φόρτωσης. Για παράδειγμα, αν χρησιμοποιείτε μόνο 2KB γραφικών, με ένα μόνο δυαδικό αρχείο μήκους 18619 θα φορτώσετε 8,5KB γραφικών, δηλαδή 6,5KB επιπλέον κενά. Αυτό στο χρόνο φόρτωσης της ταινίας μπορεί να σημαίνει σχεδόν δύο λεπτά επιπλέον χρόνο. Στο δίσκο (CPC 6128) δεν έχει σημασία, επειδή δεν χρειάζεται καθόλου χρόνο για να τα φορτώσει. Σε αυτή την περίπτωση είναι προτιμότερο να

φτιάξετε ένα ή δύο δυαδικά αρχεία που αποθηκεύουν μόνο τη μνήμη που χρησιμοποιείτε.

Για να δημιουργήσετε αυτά τα 3 αρχεία πρέπει να ακολουθήσετε τα εξής βήματα:

BHMA 1

Επεξεργαστείτε τα γραφικά με το SPEDIT και το αποτέλεσμα (το SPEDIT το στέλνει σε αρχείο .txt) αντιγράψτε το στο images_mygame.asm

BHMA 2

Επεξεργασία μουσικής με το WYZtracker

Τροποποιήστε το music_mygame.asm για να συμπεριλάβει τις μουσικές που δημιουργήθηκαν

Οι μελωδίες θα συναρμολογούνται η μία μετά την άλλη, έτσι ώστε κάθε μελωδία να ξεκινά σε διαφορετική διεύθυνση μνήμης ανάλογα με το μέγεθος της μελωδίας.

BHMA 3

Επανασυναρμολογήστε τη βιβλιοθήκη 8BP, έτσι ώστε το τμήμα της βιβλιοθήκης που επιλέγει τις μελωδίες (το player wyz) να μπορεί να γνωρίζει σε ποιες διευθύνσεις μνήμης έχουν συναρμολογηθεί (υπάρχουν και άλλες εξαρτήσεις, αλλά αυτή είναι μία από αυτές). Αφού επανασυναρμολογηθούν, θα πρέπει να αποθηκεύσετε τα πάντα με μία από αυτές τις εντολές, ανάλογα με την επιλογή συναρμολόγησης που χρησιμοποιείτε (το όνομα μπορεί να είναι ό,τι θέλετε, έχω βάλει έναν αριθμό στο τέλος για να υποδείξω κάπως την επιλογή συναρμολόγησης):

```
SAVE "yourgame0.bin",b,2350 19119
      0,
SAVE "yourgame1.bin",b,2500 17619
      0,
SAVE "yourgame2.bin",b,2480 17819
      0,
SAVE "yourgame3.bin",b,2400 18619
      0,
```

Αυτή θα είναι μια έκδοση της βιβλιοθήκης ειδικά για το παιχνίδι σας. Για παράδειγμα, η εντολή

|MUSIC,0,0,0,0,3,6 θα παίζει τη μελωδία νούμερο 3 που έχετε συνθέσει εσείς οι ίδιοι. Η μελωδία νούμερο 3 μπορεί να είναι εντελώς διαφορετική σε ένα άλλο παιχνίδι.

BHMA 4

Προγραμματίστε το παιχνίδι σας, το οποίο πρέπει πρώτα να εκτελέσει την κλήση για την εγκατάσταση των εντολών RSX, δηλαδή CALL &6b78. Και κάτι πολύ σημαντικό: μην ξεχάσετε να συμπεριλάβετε την εντολή MEMORY στην αρχή, για να αποφύγετε ότι η εκτελούμενη BASIC αποθηκεύει μεταβλητές πάνω από τη διεύθυνση όπου αρχίζει η 8BP.

MEMORY 23499 :rem χρησιμοποιήστε αυτό το MEMORY για την επιλογή 0 **MEMORY 24999 :rem χρησιμοποιήστε αυτό το MEMORY για την επιλογή 1** **MEMORY 24799 :rem χρησιμοποιήστε αυτό το MEMORY για την επιλογή 2** **MEMORY 23999 :rem χρησιμοποιήστε αυτό το MEMORY για την επιλογή 3** **MEMORY 23999 :rem χρησιμοποιήστε αυτό το MEMORY για την επιλογή 3**

Το παιχνίδι σας μπορεί να προγραμματιστεί χρησιμοποιώντας τον επεξεργαστή winape, ο οποίος είναι πολύ πιο ευέλικτος από τον επεξεργαστή AMSTRAD και μπορεί να χρησιμοποιηθεί τόσο για επεξεργασία σε assemblers (.asm) όσο και σε BASIC (.bas). Ο συντάκτης winape είναι ευαίσθητος στις λέξεις-κλειδιά και αλλάζει αυτόματα το χρώμα τους, διευκολύνοντας έτσι τον προγραμματισμό. Αφού γράψετε ένα πρόγραμμα BASIC πρέπει να το αντιγράψετε/επικολλήσετε στο παράθυρο CPC του winape. Για να το

κάνετε πιο γρήγορα, μπορείτε να ενεργοποιήσετε την επιλογή "High Speed" του winape κατά την επικόλληση, έτσι ώστε η διαδικασία επικόλλησης να είναι άμεση.

BHMA 5

Φορτώστε τα πάντα με ένα loader.bas , το οποίο θα πρέπει να κάνετε σε BASIC.

BHMA 6

Δημιουργήστε μια κασέτα ή ένα δίσκο με το παιχνίδι σας

5.3 Δημιουργήστε ένα δίσκο ή μια κασέτα με το παιχνίδι σας

5.3.1 Δημιουργία δίσκου

Για να δημιουργήσουμε ένα νέο δίσκο από το winape κάνουμε τα εξής

Αρχείο->Δίσκος Α-> νέος κενός δίσκος

Αυτό θα εμφανίσει ένα παράθυρο διαχείρισης αρχείων για να ονομάσετε το νέο αρχείο .dsk.

Αφού δημιουργηθούν, μπορείτε να αποθηκεύσετε αρχεία με την εντολή SAVE. Για να διαγράψετε ένα αρχείο χρησιμοποιείτε την εντολή "ERA" (συντομογραφία για ERASE), η οποία υπάρχει μόνο στο CPC 6128 ως μέρος του λειτουργικού συστήματος "AMSDOS" (δεν υπήρχε στο CPC464, καθώς αυτό λειτουργούσε με κασέτα).

| ERA, "game.*"

Και θα διαγραφούν

Για να φορτώσετε το παιχνίδι χρειάζεστε έναν φορτωτή που φορτώνει ένα προς ένα τα απαραίτητα αρχεία. Κάτι τέτοιο (η εντολή MEMORY εξαρτάται από την επιλογή assembly):

```
10 MEMORY 23499: "Η εντολή memory εξαρτάται από την επιλογή assembly".
15 LOAD "!pant.scr",&c000: 'μόνο αν το παιχνίδι σας έχει οθόνη
φόρτωσης
20 LOAD "yourgame0.bin"
50 RUN " !yourgame.bas"
```

Για να αποθηκεύσετε κάθε ένα από τα αρχεία πρέπει να χρησιμοποιήσετε την εντολή SAVE με τις απαραίτητες παραμέτρους, για παράδειγμα:

```
SAVE "LOADER.BAS"
SAVE "yourgame0.bin",b,23500,
19119 SAVE
"yourgame0.bin",b,23500, 19119
SAVE "yourgame.BAS"
```

Αν θέλετε να γράψετε το .dsk σε μια δισκέτα 3,5" και να το συνδέσετε σε μια εξωτερική μονάδα δισκέτας του AMSTRAD CPC 6128, θα χρειαστείτε το εύχρηστο πρόγραμμα CPCDiskXP. Από μια .dsk μπορείτε να γράψετε μια δισκέτα 3,5" σε διπλή πυκνότητα (μην ξεχάσετε να καλύψετε την οπή της δισκέτας για να "ξεγελάσετε" τον υπολογιστή).

5.3.2 Δημιουργία ταινίας με το Winape

Το πιο σημαντικό πράγμα κατά τη δημιουργία μιας ταινίας είναι να αποθηκεύσετε τα αρχεία σε αυτήν με τη σειρά με την οποία θα φορτωθούν από τον υπολογιστή. Μια ταινία δεν είναι σαν ένα δίσκο στον οποίο μπορείτε να φορτώσετε οποιοδήποτε αποθηκευμένο αρχείο, αλλά τα αρχεία είναι το ένα μετά το άλλο, οπότε πρέπει να προσέξετε ιδιαίτερα σε αυτό το σημείο.

Αν ο φορτωτής του παιχνιδιού σας είναι έτσι:

```
10 MNHMH 23499
15 LOAD "screen.scr",&c000 : rem αν το παιχνίδι σας έχει οθόνη
φόρτωσης
20 LOAD " !yourgame0.bin"
50 RUN " !yourgame.bas"
```

Το θαυμαστικό "!" είναι πολύ σημαντικό, έτσι ώστε το Amstrad να μην λάβει το μήνυμα "πατήστε play και μετά οποιοδήποτε πλήκτρο" κατά την εκτέλεση κάθε LOAD.

Πρώτα πρέπει να αποθηκεύσετε τον φορτωτή (ας πούμε ότι ονομάζεται "loader.bas"), στη συνέχεια το αρχείο "tujuego.bin" και τέλος το αρχείο "tujuego.BAS".

Για να δημιουργήσετε ένα ".wav" ή από το winape

file->tape->press record

Στη συνέχεια, θα εμφανιστεί ένα μενού διαχείρισης αρχείων, ώστε να αποφασίσουμε πώς θα ονομάσουμε το αρχείο ".wav".

Εάν βρίσκεστε σε λειτουργία CPC 6128, τότε στη συνέχεια πρέπει να εκτελέσετε από τη BASIC

| **TAPE**

Και μετά

TAXYTHTA EΓΓΡΑΦΗΣ 1

Με αυτή την εντολή αυτό που θα έχουμε κάνει είναι να πούμε στο AMSTRAD να κάνει εγγραφή με 2000 baud. Αυτό θα μειώσει το χρόνο φόρτωσης. Αν δεν εκτελέσετε αυτή την εντολή, η εγγραφή θα γίνει στα 1000 baud, που είναι ασφαλέστερο αλλά πολύ πιο αργό.

SAVE "LOADER.BAS"

Θα εμφανιστεί ένα μήνυμα που θα σας λέει να πατήσετε rec&play και, στη συνέχεια, πατήστε "ENTER". Στη συνέχεια, αποθηκεύστε κάθε αρχείο:

```
SAVE "yourgame0.bin",b,23500,  
19119 SAVE  
"yourgame0.bin",b,23500, 19119  
SAVE "yourgame.BAS"
```

Τέλος, πρέπει να κάνουμε μια τελευταία πράξη για να κλείσει το αρχείο το winape.

file->remove tape

Αφού κάνετε την "αφαίρεση ταινίας", το αρχείο θα αποκτήσει το μέγεθός του (αν δεν το κάνετε, μπορείτε να δείτε ότι στο δίσκο του υπολογιστή σας το αρχείο δεν μεγαλώνει και αυτό συμβαίνει επειδή δεν έχει μεταφερθεί στο δίσκο).

Για να φορτώσετε το παιχνίδι, αν είστε σε ένα CPC6128

| **TAPE**

RUN ""

Για να επαναχρησιμοποιήσετε το δίσκο

| **DISC**

Αν θέλετε να αποθηκεύσετε μια οθόνη φόρτωσης, ανατρέξτε στο Παράρτημα I σχετικά με την οργάνωση της μνήμης βίντεο, όπου εξηγώ πώς να το κάνετε.

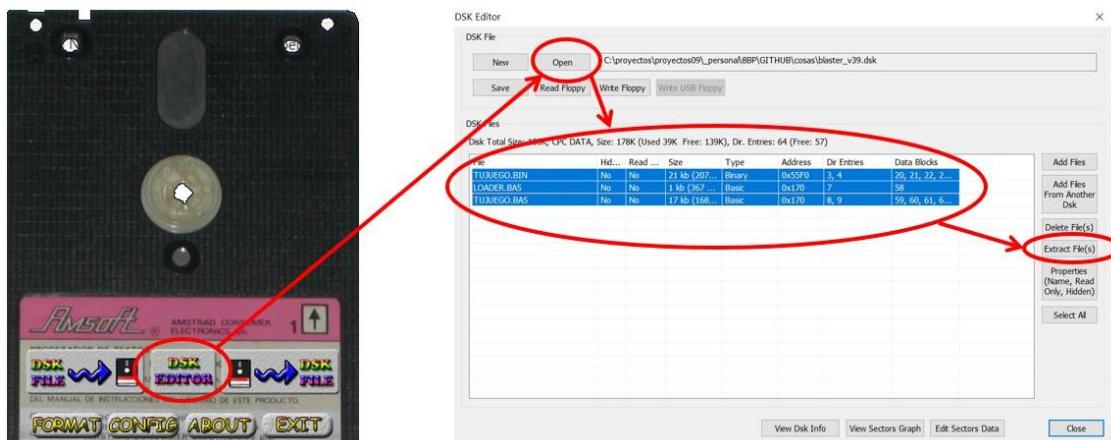
5.3.3 Φτιάξτε μια ταινία εύκολα με τα CPCDiskXP, 2cdt και tape2wav

Το Winape είναι ένα εξαιρετικό εργαλείο για προγραμματισμό και εξομοίωση. Ωστόσο, έχει έναν μικρό περιορισμό: δεν σας επιτρέπει να φτιάξετε μια κασέτα σε μορφή cdt, μόνο σε μορφή wav.

Υπάρχει ένας πολύ γρήγορος και αξιόπιστος τρόπος που θα σας επιτρέψει να φτιάξετε αρχεία .cdt και wav για το χρειάζεστε τα εργαλεία CPCDiskXP, 2cdt και tape2wav.

Ας ξεκινήσουμε από την υπόθεση ότι έχετε δημιουργήσει ένα .dsk με τα αρχεία σας. Με αυτό, πρέπει να κάνετε τα ακόλουθα βήματα:

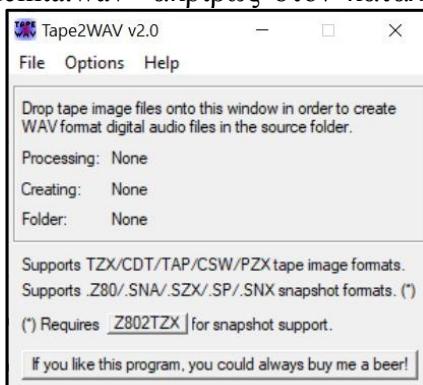
- 1) Πρώτα απ' όλα, με το εργαλείο CPCDiskXP μπορείτε να ανοίξετε το .dsk και να εξάγετε τα απαραίτητα αρχεία για το παιχνίδι σας (το "loader.bas", το "tujuego.bin" και το "tujuego.bas"). Για να το κάνετε αυτό, απλώς κάνετε κλικ στο "disk edition", στη συνέχεια στο "open", ανοίγετε το .dsk σας, επιλέγετε τα αρχεία και τέλος κάνετε κλικ στο "extract files". Μόλις το κάνετε αυτό, θα έχετε τα αρχεία στο σύστημα αρχείων των windows.



- 2) Στη συνέχεια, χρησιμοποιείτε το εργαλείο 2cdt για να εγγράψετε τα αρχεία, ένα προς ένα, σε ένα αρχείο .cdt. Οι εντολές θα είναι οι εξής:

```
2cdt.exe -n -s 1 -r "LOADER.bas" "loader.bas" tucinta.cdt
2cdt.exe -b 2000 -r "tujuego0.bin" "tujuego0.bin" tucinta.cdt
2cdt.exe -b 2000 -r "tujuego.bas" "tujuego.bas" tucinta.cdt
```

- 3) Τώρα έχετε δημιουργήσει το αρχείο tucinta.cdt. Αν θέλετε επίσης να έχετε ένα αρχείο .wav για να μπορείτε να το φορτώσετε σε ένα πραγματικό CPC 464, μπορείτε να χρησιμοποιήσετε το εργαλείο tape2wav. Απλώς το ξεκινάτε και σύρετε το αρχείο .cdt με το ποντίκι στο εργαλείο. Το tape2wav θα δημιουργήσει αμέσως ένα αρχείο "tucinta.wav" ακριβώς στον κατάλογο όπου βρισκόταν το

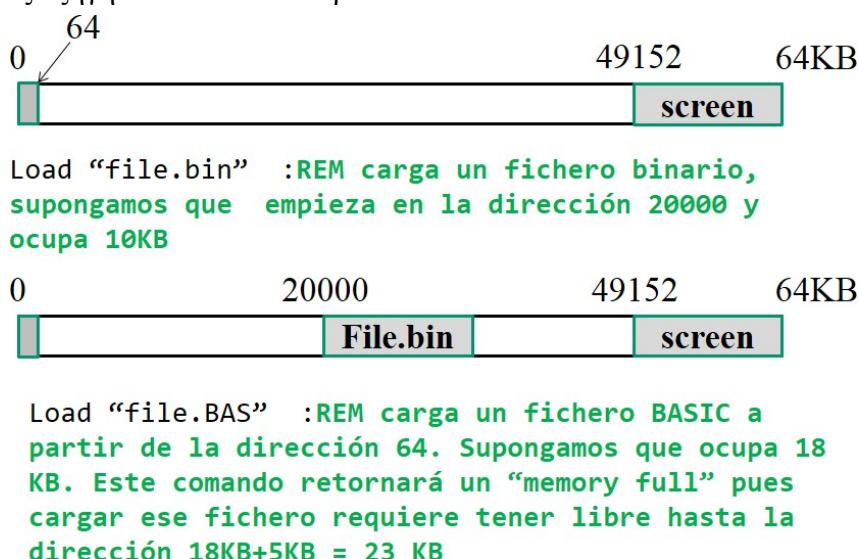


"tucinta.cdt".

5.3.4 Αντιμετώπιση προβλημάτων LOAD και MEMORY

Πριν από τη φόρτωση ενός αρχείου BASIC, το Amstrad βεβαιώνεται ότι θα έχει διαθέσιμο χώρο για την εκτέλεσή του. Το πρόγραμμα BASIC μπορεί να χρησιμοποιεί μόνο 1KB μεταβλητών, αλλά το Amstrad δεν το γνωρίζει αυτό, οπότε είναι πιο συντηρητικό και απαιτεί να έχετε επιπλέον 5KB κενής μνήμης. Αυτά τα 5KB μπορεί να φαίνονται υπερβολικά, αλλά το Amstrad δεν γνωρίζει εκ των προτέρων πόσες μεταβλητές πρόκειται να δηλώσετε στο πρόγραμμά σας και προτιμά να έχει πολύ ελεύθερο χώρο για την αποθήκευση των μεταβλητών παρά να υπολείπεται και να αποτύχει το πρόγραμμα.

Αυτό σημαίνει ότι αν έχετε προηγουμένως φορτώσει ένα ή περισσότερα δυαδικά αρχεία (με γραφικά, βιβλιοθήκη 8BP ή οτιδήποτε άλλο) και έχετε αφήσει 20 KB ελεύθερα, τότε δεν θα μπορείτε να φορτώσετε ένα παιχνίδι BASIC των 20 KB αλλά το πολύ ένα παιχνίδι των 15 KB. Άλλα μην ανησυχείτε, υπάρχουν διάφοροι τρόποι για να το λύσετε αυτό. Θα σας εξηγήσω τον ευκολότερο



Σχ. 13 Το πρόβλημα LOAD

Η λύση είναι να δημιουργήσετε ένα αρχείο "loader.bas" που αλλάζει τη MNHMH. Ας υποθέσουμε ότι έχετε δυαδικά δεδομένα από τη διεύθυνση 20.000 και το πρόγραμμά σας καταλαμβάνει 18KB, αφήνοντας λιγότερο από 5KB χώρο. Το μόνο που έχετε να κάνετε είναι: "Η μνήμη του αρχείου είναι πολύ μεγάλη:

```

10 MNHMH 19999
20 LOAD "!game.bin": rem φορτώνει δεδομένα από 20000
30 CLEAR: MEMORY 23000 : rem ώστε να σας δώσουμε περισσότερη
ελεύθερη μνήμη RAM. 40 RUN "!game.bas": rem η πρώτη γραμμή του
game.bas πρέπει να έχει μνήμη 19999
  
```

Αυτή η μέθοδος είναι πολύ απλή και 100% αξιόπιστη, διότι, παρόλο που αφήνετε τιμήματα των δυαδικών δεδομένων απροστάτευτα κατά τη δύρκεια της φόρτωσης της BASIC, το πρώτο πράγμα που κάνετε στη BASIC είναι να εκτελέσετε τη MEMORY, οπότε προστατεύεται ξανά πριν δημιουργήσετε οποιεσδήποτε μεταβλητές.

Ένα άλλο τυπικό πρόβλημα που σχετίζεται με το σφάλμα MEMORY FULL εμφανίζεται όταν φορτώνουμε ένα πρόγραμμα και στη μέση της εκτέλεσης το σταματάμε (πατώντας δύο φορές ESC). Είναι πιθανό να έχουμε MEMORY FULL όταν προσπαθούμε να κάνουμε πράγματα όπως η πρόσβαση στο δίσκο με μια εντολή

CAT.

```
Break in 420
Ready
CAT
Memory full
Ready
```

Αυτό οφείλεται στο γεγονός ότι το πρόγραμμα BASIC μπορεί να καταναλώσει πολλή μνήμη RAM μεταβλητών. Η διακοπή δεν σημαίνει ότι οι μεταβλητές έχουν εξαφανιστεί από το πρόγραμμα, στην πραγματικότητα, εξακολουθούν να υπάρχουν και μπορείτε ακόμη και να τις εκτυπώσετε για να δείτε την τιμή τους. Αυτό που θα κάνουμε σε αυτή την περίπτωση είναι απλά να εκτελέσουμε την εντολή **CLEAR**, η οποία απελευθερώνει τη μνήμη αυτών των μεταβλητών και στη συνέχεια την εντολή CAT μας (ή αυτή που θέλουμε).

```
Break in 420
Ready
CAT
Memory full
Ready
CLEAR
Ready
CAT

Drive A: user 0
LOADER :BAK 1K SP :BAS 18K
LOADER :BAS 1K SP :BIN 19K
SP :BAK 18K SP :SCR 17K

104K free
Ready
```

Αν έχετε φτιάξει ένα πρόγραμμα BASIC τόσο μεγάλο που δεν έχετε 5KB ελεύθερα μεταξύ του προγράμματος BASIC και του συναρμολογημένου δυαδικού αρχείου, τότε λογικά θα λάβετε επίσης το σφάλμα **MEMORY FULL** όταν πάτε να αποθηκεύσετε το παιχνίδι σας στο δίσκο.

Αν προσπαθήσετε να αποθηκεύσετε το δυαδικό αρχείο, θα λάβετε το σφάλμα **MEMORY FULL**, αλλά είναι πολύ εύκολο να το διορθώσετε. Απλά μην φορτώσετε την καταχώριση BASIC στον εξομοιωτή σας και μην εκτελέσετε καμία εντολή MEMORY. Όταν συναρμολογήσετε με το winape το αρχείο "make_all.asm" θα έχετε όλα τα γραφικά, τη μουσική και τη βιβλιοθήκη 8BP συναρμολογημένα στη μνήμη του Amstrad. Στη συνέχεια, εκτελέστε την εντολή SAVE και θα λειτουργήσει. Η εντολή για να αποθηκεύσετε τα πάντα σε ένα ενιαίο δυαδικό αρχείο εξαρτάται από την επιλογή συναρμολόγησης που βάλατε στο αρχείο **make_all_mygame.asm**, και θα είναι μία από αυτές:

```
SAVE "yourgame0.bin",b,2350 19119
      0,
SAVE "yourgame1.bin",b,2500 17619
      0,
SAVE "yourgame2.bin",b,2480 17819
      0,
SAVE "yourgame3.bin",b,2400 18619
      0,
```

Ωστόσο, αν έχετε αποθηκεύσει δυαδικά πράγματα (επιπλέον γραφικά, χάρτες κ.λπ.)

κάτω από την αρχική διεύθυνση 8BP, θα πρέπει να το λάβετε υπόψη σας. Για παράδειγμα, αν το παιχνίδι σας αρχίζει να χρησιμοποιεί μνήμη στη διεύθυνση 20000, η εντολή θα είναι (σημειώστε ότι έχω αυξήσει το μήκος ώστε να καταλαμβάνει από 20000 έως 42619)

```
SAVE "yourgame.bin",b,20000, 22619
```

Μπορείτε τώρα να αντιγράψετε και να επικολλήσετε το πρόγραμμα BASIC στον εξόμοιωτή. Αφού αντιγράψετε, μην εκτελέσετε καμία εντολή **MEMORY** και αποθηκεύστε το αρχείο .BAS στο δίσκο.

Καθώς δεν έχετε εκτελέσει ακόμη κάποια εντολή **MEMORY** σε αυτό το σημείο, το Amstrad "νομίζει" ότι έχετε περισσότερα από 5KB ελεύθερα πάνω από το πρόγραμμα BASIC και δεν θα μας δώσει το μήνυμα **MEMORY FULL**. Μόλις εκτελέσετε την εντολή **MEMORY** (για παράδειγμα, αν τα δυαδικά δεδομένα σας ξεκινούν από το 20000 θα είναι ένα **MEMORY 19999** αντί για 23999) το Amstrad θα ελέγξει αν έχετε 5KB ελεύθερα μεταξύ του προγράμματος BASIC και της διεύθυνσης **MEMORY** και αν δεν υπάρχουν αρκετά, θα σας δώσει ένα σφάλμα κατά την εκτέλεση της εντολής **SAVE**, ακόμα και αν το παιχνίδι λειτουργεί. Εάν κατά τη διάρκεια της εκτέλεσης το παιχνίδι σας προσπαθήσει να καταναλώσει περισσότερο χώρο από αυτόν που έχει ελεύθερο στη διεύθυνση **MEMORY**, θα σταματήσει και θα δώσει το σφάλμα **MEMORY FULL**.

6 Βιβλιοθήκη, μουσική και συναρμολόγηση γραφικών

Αυτό το κεφάλαιο περιγράφει λίγο περισσότερο τι συμβαίνει όταν εκτελείτε το αρχείο "make_all" και θα σας επιτρέψει να κατανοήσετε καλύτερα την όλη διαδικασία, **αν και αν θέλετε να αρχίσετε να μαθαίνετε πώς να προγραμματίζετε με την 8BP, μπορείτε να το προσπεράσετε και να επιστρέψετε εδώ αργότερα**, όταν θέλετε να κατανοήσετε καλύτερα πού να τοποθετήσετε τα γραφικά και τη μουσική και πώς να συναρμολογήσετε τη βιβλιοθήκη με αυτά.

Αυτό συμβαίνει επειδή, για παράδειγμα, το πρόγραμμα αναπαραγωγής μουσικής είναι ενσωματωμένο στη βιβλιοθήκη και πρέπει να γνωρίζει από πού ξεκινάει κάθε τραγούδι (διεύθυνση μνήμης), οπότε είναι απαραίτητο να επανασυναρμολογήσετε και να αποθηκεύσετε την έκδοση της βιβλιοθήκης που αφορά το παιχνίδι σας, καθώς και το συναρμολογημένο αρχείο γραφικών και το συναρμολογημένο αρχείο μουσικής.

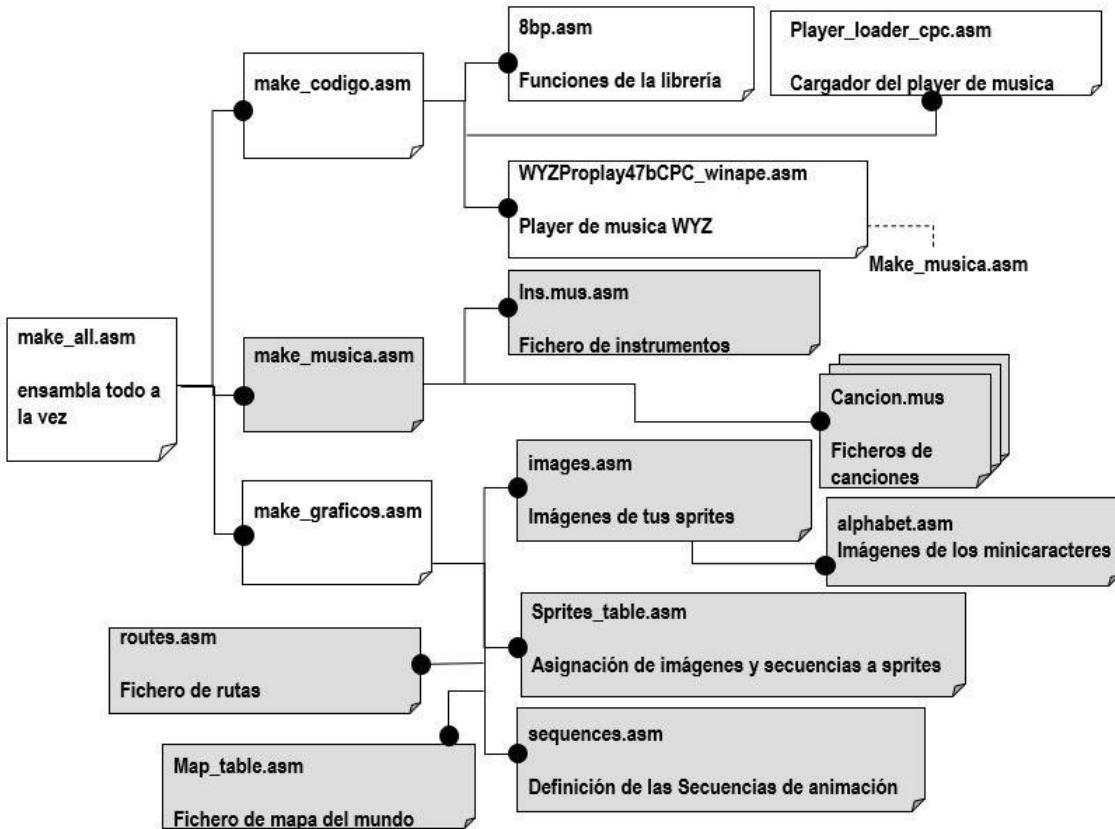
Οπως εξήγησα στην ενότητα "Βήματα", αυτή θα είναι μια έκδοση της βιβλιοθήκης ειδικά για το παιχνίδι σας. Για παράδειγμα, η εντολή **|MUSIC,0,0,3,6** θα παίξει τη μελωδία νούμερο 3 που έχετε συνθέσει εσείς οι ίδιοι. Η μελωδία νούμερο 3 μπορεί να είναι εντελώς διαφορετική σε ένα άλλο παιχνίδι. Το ίδιο ισχύει και για τα δεδομένα στο αρχείο οργάνων. Υπάρχουν ορισμένες εξαρτήσεις μεταξύ του κώδικα του music player και των διευθύνσεων όπου συγκεντρώνονται τα δεδομένα των οργάνων και οι μελωδίες.

Είναι πολύ απλό, αλλά πρέπει να κατανοήσετε τη δομή της βιβλιοθήκης για να το κάνετε, δηλαδή τη δομή των αρχείων .asm που πρέπει να χειριστείτε και τις εξαρτήσεις τους.

Στο παρακάτω διάγραμμα παρουσιάζονται όλα τα αρχεία .asm ενός παιχνιδιού που χρησιμοποιεί την 8BP καθώς και οι μεταξύ τους εξαρτήσεις. Στο σχήμα, **με γκρι χρώμα είναι τα αρχεία που πρέπει να επεξεργαστείτε**, όπως:

- τα τραγούδια και το αρχείο οργάνων, τα οποία δημιουργείτε με το WYZtracker
- το αρχείο **make_musica** όπου δηλώνετε ποια αρχεία ".mus" πρόκειται να συναρμολογηθούν
- το αρχείο εικόνας που δημιουργείτε με το SPEDIT
- τον πίνακα sprite όπου αντιστοιχίζετε εικόνες στα sprites (αν και αυτό δεν είναι απολύτως απαραίτητο, καθώς έχετε την εντολή **|SETUPSP**).
- τον πίνακα ακολουθιών, όπου ορίζετε ποιες εικόνες αποτελούν μια ακολουθία,
- ο παγκόσμιος χάρτης: όπου μπορείτε να ορίσετε έως και 64 στοιχεία που συνθέτουν τον κόσμο.
- το αρχείο paths: όπου ορίζετε τις διαδρομές των sprites που θέλετε να χρησιμοποιήσετε.
- το αρχείο alphabet.asm: αν θέλετε να δημιουργήσετε ένα αλφάριθμο διαφορετικό από το τυπικό αλφάριθμο της 8BP

Μπορείτε να συναρμολογήσετε τα πάντα ανοίγοντας το αρχείο "**make_all.asm**" και πατώντας "assemble" στο μενού του Winape. Στη συνέχεια, μπορείτε να χρησιμοποιήσετε την εντολή SAVE για να αποθηκεύσετε τις εικόνες, τη μουσική και τη βιβλιοθήκη 8BP σε διαφορετικά δυαδικά αρχεία ή σε ένα μόνο, όπως είδαμε.



Σχ. 14 Αρχεία συναρμολόγησης

Αν αλλάζετε μόνο τα γραφικά, μπορείτε να τα συναρμολογήσετε ξεχωριστά, επιλέγοντας το αρχείο "make_graphics.asm" και πατώντας assemble.

Αν αλλάζετε τη μουσική, πρέπει να επανασυναρμολογήσετε τον κώδικα της βιβλιοθήκης, επειδή υπάρχει μια εξάρτηση μεταξύ του κώδικα και των τραγουδιών, επειδή ο κώδικας πρέπει να γνωρίζει από πού αρχίζει κάθε τραγούδι. Έτσι, αν αλλάζετε ή προσθέσετε τραγούδια πρέπει να συναρμολογήσετε με το make_all.asm. Έχω αποτυπώσει αυτή την εξάρτηση με μια διακεκομμένη γραμμή μεταξύ του player και του αρχείου make_musica.asm.

Ίσως χρειαστεί να συναρμολογήσετε κάτι αλλο, όπως έναν χάρτη αγωνιστικών διαδρομών που χρησιμοποιεί τις εικόνες σας. Σε αυτή την περίπτωση, προσθέστε το στο αρχείο "Make_graphics.asm" ώστε να συναρμολογηθεί μετά το images.asm. Η σειρά συναρμολόγησης είναι σημαντική. Πρώτα πρέπει να συναρμολογήσετε τις εικόνες, να τους αντιστοιχίσετε ετικέτες και μετά μπορείτε να συναρμολογήσετε τους χάρτες ή τις πίστες που χρησιμοποιούν αυτές τις ετικέτες.

6.1 Make_all.asm

Αυτό είναι το αρχείο που επιτρέπει τη συναρμολόγηση των πάντων. Εσωτερικά καλεί τρία αρχεία που συναρμολογούν τον κώδικα της βιβλιοθήκης και του music player, τα τραγούδια και τα γραφικά.

; Makefile για βιντεοπαιχνίδια που χρησιμοποιούν ισχύ 8bit
Αν αλλάζετε ένα μέρος του, το μόνο που χρειάζεται να κάνετε είναι να συναρμολογήστε το αντίστοιχο make
για παράδειγμα, μπορείτε να συναρμολογήσετε το make_graphics αν αλλάζετε σχέδια

ΑΠΟ ΤΟ V42 ΥΠΑΡΧΟΥΝ "ΕΠΙΛΟΓΕΣ" ΣΥΝΑΡΜΟΛΟΓΗΣΗΣ.

```

; ASSEMBLING_OPTION = 0 --> όλες οι διαθέσιμες εντολές.

; ASSEMBLING_OPTION = 1 --> για παιχνίδια λαβύρινθου. MNHMH 25000
; διαθέσιμες οι εντολές |LAYOUT, |COLAY
;

; ASSEMBLING_OPTION = --> για παιχνίδια κύλισης, MNHMH 24800
; διαθέσιμες οι εντολές |MAP2SP, |UMA
;

; ASSEMBLING_OPTION = --> για ψευδοτρισδιάστατα παιχνίδια , MNHMH
; 24000
; διαθέσιμη εντολή |3D
;

; ASSEMBLING_OPTION = --> μελλοντική χρήση

let ASSEMBLING_OPTION = 0
-----CÓDIGO-----
;perílambaránει τη βιβλιοθήκη 8bp και το music
playerWYZ διαβάστε το "make_codigo_mygame.asm".

-----MUSICA-----
διαβάστε το
"make_musica_mygame.asm",
περιλαμβάνει τα τραγούδια.

----- ΓΡΑΦΗΜΑΤΑ -----
Αντό το μέρος περιλαμβάνει εικόνες και ακολουθίες κινουμένων
σχεδίων.

; και ο πίνακας sprite αρχικοποιείται με αυτές τις εικόνες και τις
ακολούθιες που διαβάζονται στο "make_graficos_mygame.asm"
Καθε ενα από αυτά τα τρία αρχεία είναι υπεύθυνο για τη συναρμολόγηση διαφορετικών
πραγμάτων και, για παράδειγμα, το αρχείο γραφικών καλεί άλλα αρχεία όπως το αρχείο
εικόνας, το αρχείο ακολουθίας, το αρχείο διαδρομής και τον παγκόσμιο χάρτη.
```

Χρησιμοποιείτε πάντα την επιλογή συναρμολόγησης που σας δίνει τη μεγαλύτερη ελεύθερη μνήμη. Αν το παιχνίδι σας είναι ένα παιχνίδι λαβύρινθος, χρησιμοποιήστε την επιλογή 1, και αν είναι ένα παιχνίδι κύλισης, χρησιμοποιήστε την επιλογή 2. Αν είναι ένα ψευδο3D παιχνίδι, χρησιμοποιήστε την επιλογή 3. Σε γενικές γραμμές δεν σας συνιστώ να χρησιμοποιήσετε την επιλογή 0, διότι είναι αυτή που σας δίνει τη λιγότερη ελεύθερη μνήμη και πιθανότατα θα μπορέσετε να χρησιμοποιήσετε μια από τις άλλες επιλογές.

6.2 Δομή του αρχείου εικόνας

```
IMAGE_LIST
    Lista de imágenes en orden de numeración (comenzando en 16)

-BEGIN_ALPHABET
    Carga del fichero de alfabeto
-END_ALPHABET

-BEGIN_FLIP_IMAGES
    listado de imágenes flipeadas
-END_FLIP_IMAGES

-BEGIN_BG_IMAGES
    listado de imágenes de fondo (“background images”)
-END_BG_IMAGES

Definición (todos los dibujos byte a byte) de imágenes

-BEGIN_ZOOM_IMAGES
    Definición de imágenes de tipo zoom
-END_ZOOM_IMAGES

-BEGIN_3D_SEGMENTS
    Definicion de imágenes de tipo Segmento
-END_3D_SEGMENTS

-END_GRAPH
```

Σχ. 15 Δομή του αρχείου εικόνας

6.3 Δομή αρχείου ακολουθίας κινούμενων σχεδίων

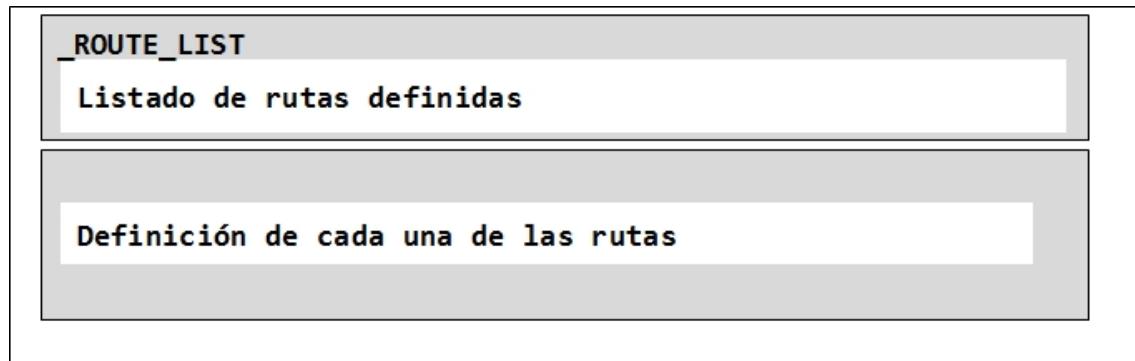
```
_SEQUENCES_LIST
    Definición de secuencias de animación

_MACRO_SEQUENCES
    Definición de macro secuencias de animación

-END_ALPHABET
```

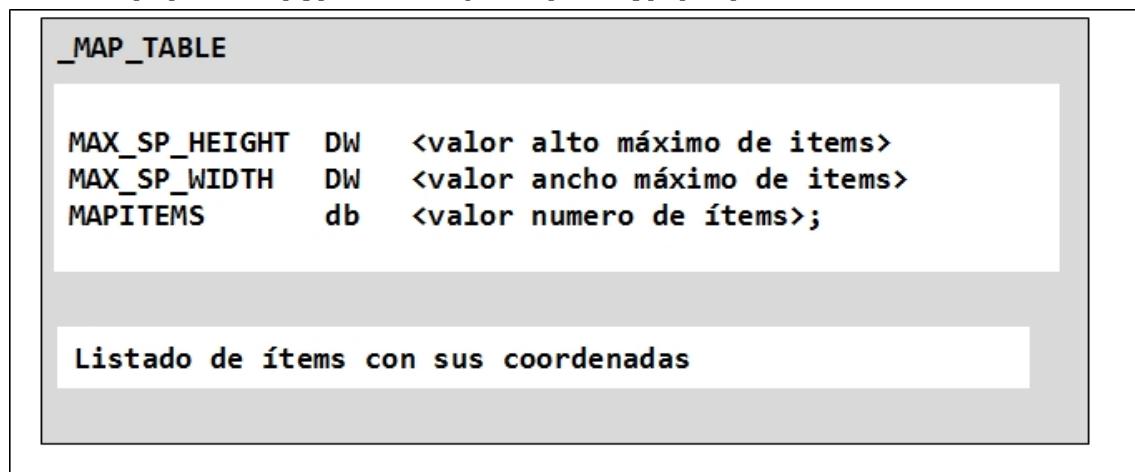
Σχ. 16 Δομή αρχείου ακολουθίας κινούμενων σχεδίων

6.4 Δομή του αρχείου δρομολόγησης



Σχ. 17 Δομή του αρχείου δρομολόγησης

6.5 Δομή του αρχείου παγκόσμιου χάρτη



Σχ. 18 Δομή αρχείου παγκόσμιου χάρτη

7 Κύκλος παιχνιδιού

Ένα βιντεοπαιχνίδιο arcade, πλατφόρμας, περιπέτειας έχει γενικά έναν παρόμοιο τύπο δομής, όπου ορισμένες λειτουργίες επαναλαμβάνονται κυκλικά σε αυτό που θα ονομάσουμε "κύκλο παιχνιδιού".

Σε κάθε κύκλο παιχνιδιού θα ενημερώνουμε τις θέσεις των sprites και θα εκτυπώνουμε τα sprites στην οθόνη, έτσι ώστε ο αριθμός των κύκλων παιχνιδιού που εκτελούνται ανά δευτερόλεπτο να ισούται με τα "καρέ ανά δευτερόλεπτο" (FPS) του παιχνιδιού. Ο παρακάτω ψευδοκώδικας περιγράφει τη βασική δομή ενός παιχνιδιού

INICIO

```
Inicialización de variables globales: vidas, etc  
Espera a que el usuario pulse tecla de comienzo de juego  
GOSUB pantalla1  
GOSUB pantalla 2  
...  
GOSUB pantalla N  
GOTO INICIO
```

Código de PANTALLA N (siendo N cualquier pantalla)

```
Inicialización de coordenadas de enemigos y personaje  
Pintado de la pantalla (layout), si procede
```

BUCLE PRINCIPAL (ciclo del juego en esta pantalla)

```
Lógica de personaje : Lectura de teclado y actualización de  
coordenadas del personaje y si procede, actualización de su  
secuencia de animación
```

```
Ejecución de lógica de enemigo 1  
Ejecución de lógica de enemigo 2  
...  
Ejecución de lógica de enemigo n
```

```
Impresión de todos los sprites  
Condición de salida de esta pantalla (IF ... THEN RETURN)  
GOTO BUCLE PRINCIPAL
```

Σχ. 19 Βασική δομή ενός παιχνιδιού

Εάν η λογική των εχθρών είναι πολύ βαριά λόγω υπερβολικά πολλών εχθρών ή πολύ σύνθετη, αυτό θα καταναλώνει περισσότερο χρόνο σε κάθε κύκλο παιχνιδιού και, επομένως, ο αριθμός των κύκλων ανά δευτερόλεπτο θα μειωθεί. Προσπαθήστε να μην πέσετε κάτω από τα 10fps για να διατηρήσει το παιχνίδι ένα αποδεκτό επίπεδο δράσης.

7.1 Πώς να μετρήσετε τα FPS του κύκλου του παιχνιδιού σας

Για να ξέρετε αν το παιχνίδι σας έχει αποδεκτό επίπεδο δράσης, δεν υπάρχει τίποτα καλύτερο από το να το παίξετε και αν σας αρέσει, τότε θα είναι μια χαρά. Ωστόσο, μπορεί να θέλετε να μετρήσετε ακριβώς πόσα καρέ ανά δευτερόλεπτο είναι ικανό να παράγει το παιχνίδι σας, διότι τότε μπορείτε να πάρετε αποφάσεις στη λογική του προγράμματός σας και να μετρήσετε πόσο αυτές οι προγραμματιστικές αποφάσεις το βλάπτουν ή το ωφελούν.

Αυτό που θα κάνουμε για να μετρήσουμε είναι απλώς να σημειώσουμε τη χρονική στιγμή πριν από την έναρξη του πρώτου κύκλου παιχνιδιού, στην αρχή του "κώδικα Νοθόνης". Στη συνέχεια, θα σημειώσουμε τον χρόνο μετά από μερικούς κύκλους παιχνιδιού και θα κάνουμε μια απλή διαίρεση. Ας το δούμε βήμα προς βήμα:

A=TIME : Η γραμμή αυτή αποθηκεύει στη μεταβλητή A τον χρόνο σε 1/300 κλάσματα του δευτερολέπτου.

Ο αριθμός που πρέπει να αποθηκευτεί στο A μπορεί να είναι πολύ μεγάλος, στην πραγματικότητα μπορεί να είναι μεγαλύτερος από αυτόν που μπορεί να αποθηκεύσει μια ακέραια μεταβλητή όπως το "A". Για να μην παράγει σφάλμα η ανάθεση, είναι βολικό να μηδενίσετε το χρονόμετρο του AMSTRAD, το οποίο εκκινείται κάθε φορά που εκκινείται η μηχανή. Για να τον μηδενίσετε, πριν αναθέσετε τη μεταβλητή "A", απλά εκτελέστε:

Σε ένα CPC 6128

POKE &b8b4,0: POKE &b8b5,0: POKE &b8b6,0: POKE &b8b7,0

Σε ένα CPC 464

POKE &b187,0: POKE &b188,0: POKE &b189,0: POKE &b18a,0

Για να διακρίνετε σε ποιο μηχάνημα βρίσκεται το πρόγραμμά σας, πρέπει να απενεργοποιήσετε τη μουσική και να συμβουλευτείτε μια διεύθυνση με το PEEK.

**| MUSIC: Αν peek(&39)=57 τότε Model=464 αλλιώς
model=6128 Αν model=464 τότε ...**

Με αυτόν τον τρόπο θα έχετε μηδενίσει τις διευθύνσεις μνήμης στις οποίες το AMSTRAD αποθηκεύει το χρονόμετρο. Στη συνέχεια, εκτελούμε όσους κύκλους παιχνιδιού θέλουμε και ελέγχουμε σε ποιον κύκλο βρισκόμαστε με τη μεταβλητή "cycle", την οποία θα αυξάνουμε κατά μία μονάδα σε κάθε κύκλο. Αφού βγούμε από τη συγκεκριμένη φάση ή οθόνη, εκτελούμε την εντολή :

FPS= κύκλος * 300/ (ΧΡΟΝΟΣ - Α)

Και τώρα έχουμε τα FPS του παιχνιδιού μας. Θα τα βάλω όλα στη σειρά για εσάς παρακάτω:

Rem υποθέτουμε ότι βρισκόμαστε σε ένα 6128

POKE &b8b4,0: POKE &b8b5,0: POKE &b8b6,0: POKE &b8b7,0 A=TIME

<εδώ μεταβείτε στο το πρόγραμμα που τρέχει το κύκλος του κύκλο, συμπεριλαμβανομένου του κύκλου=κύκλος+1 >

Θα λάβουμε εδώ μετά την κατάσταση εξόδου της οθόνης

FPS= cycle * 300/ (TIME - A)

PRINT "FPS =";FPS

Για να γίνει κατανοητό: Ο χρόνος που πέρασε από την έναρξη του προγράμματος μέχρι την ολοκλήρωσή του είναι ΧΡΟΝΟΣ-Α εκφρασμένος σε 1/300 κλάσματα του δευτερολέπτου. Για να τον μετατρέψετε σε δευτερόλεπτα, διαιρέστε τον με το 300.

Δευτερόλεπτα= (ΧΡΟΝΟΣ - Α) /300

Εάν έχουν εκτελεστεί οι κύκλοι σε αυτά τα δευτερόλεπτα (π.χ.), τότε έχει διαρκέσει ένας κύκλος: **Tc= 300*(ΧΡΟΝΟΣ-Α)/n**

Και ο αριθμός των κύκλων που μπορούν να εκτελεστούν σε ένα δευτερόλεπτο (το FPS) είναι το αντίστροφο, δηλαδή **FPS = 1/Tc = n*300/(TIME-A)**.

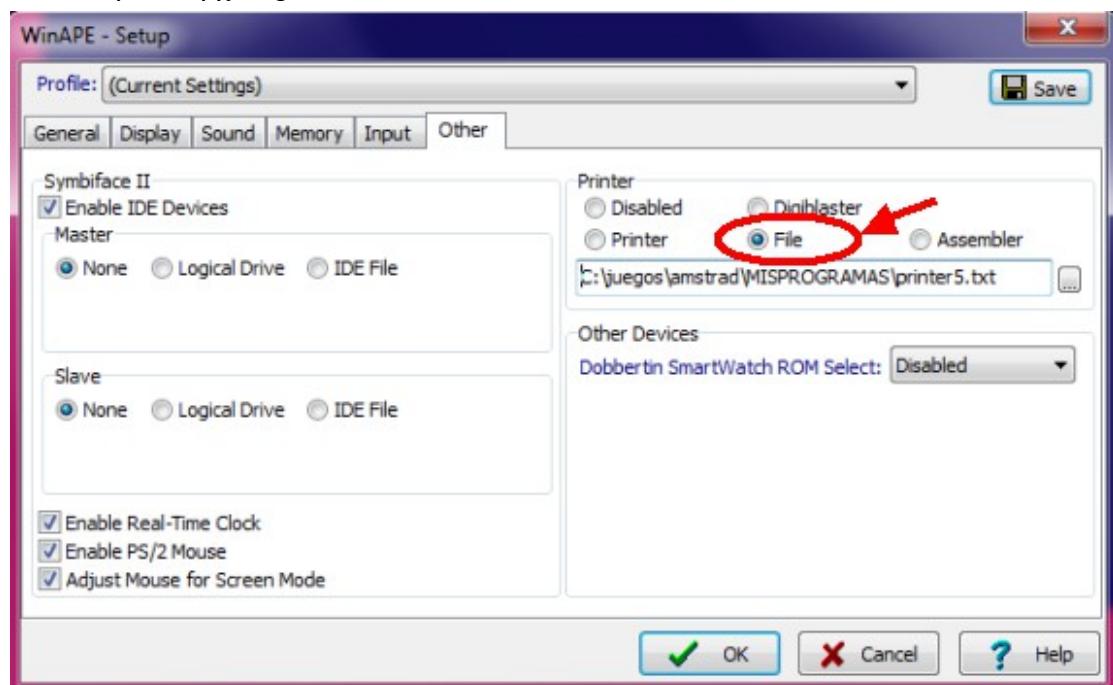
8 Sprites

8.1 Επεξεργασία sprites με το SPEDIT και συναρμολόγησή τους

Το Spedit (Simple Sprite Editor) είναι ένα εργαλείο που θα σας επιτρέψει να δημιουργήσετε τις δικές σας εικόνες χαρακτήρων και εχθρών και να τις χρησιμοποιήσετε στα BASIC προγράμματά σας.

Το Spedit είναι φτιαγμένο σε BASIC και είναι πολύ απλό, οπότε μπορείτε να το τροποποιήσετε ώστε να κάνει πράγματα που δεν προβλέπονται και σας ενδιαφέρουν. Τρέχει στο Amstrad CPC, αν και έχει σχεδιαστεί για χρήση από τον εξομοιωτή winape.

Το πρώτο πράγμα που πρέπει να κάνετε είναι να ρυθμίσετε το winape ώστε να εξάγει την έξοδο του εκτυπωτή σε ένα αρχείο. Σε αυτό το παράδειγμα έβαλα την έξοδο του εκτυπωτή στο αρχείο printer5.txt.



Εικ. 20 Επανακατεύθυνση του εκτυπωτή CPC σε ένα αρχείο με το Winape

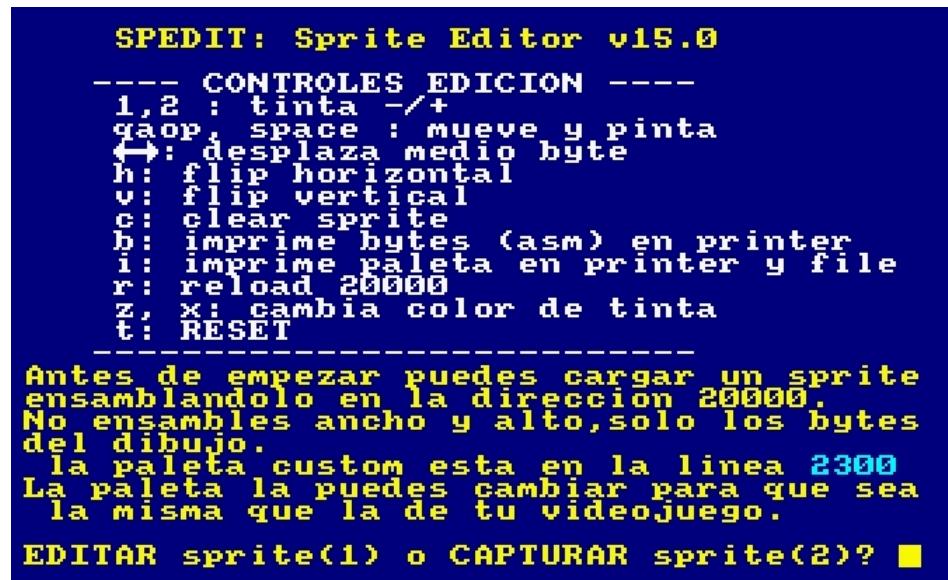
Όταν εκτελέσετε το SPEDIT θα εμφανιστεί το ακόλουθο μενού, όπου μπορείτε να επιλέξετε μεταξύ της επεξεργασίας ενός Sprite ή της σύλληψης ενός Sprite από ένα αρχείο εικόνας (.scr).

Η καταγραφή Sprite είναι διαθέσιμη από το SPEDIT V14. Σε περίπτωση που θέλετε να συλλάβετε ένα sprite, θα πρέπει να έχετε ένα αρχείο εικόνας (.scr) στο δίσκο, το οποίο είναι απλώς ένα δυαδικό αρχείο με 16384 bytes. Μπορεί να είναι μια εικόνα από ένα παιχνίδι που έχετε συλλάβει, όπου υπάρχουν εικόνες χαρακτήρων που σας αρέσουν και δεν θέλετε να ξοδέψετε χρόνο για την επεξεργασία τους.

Σε περίπτωση που επιλέξετε να επεξεργαστείτε ένα Sprite, το πρόγραμμα σας ζητάει την παλέτα που θα χρησιμοποιήσετε. Μπορείτε να επιλέξετε μια προεπιλεγμένη παλέτα

ή μια δική σας παλέτα που θέλετε να ορίσετε. Μπορείτε επίσης να χρησιμοποιήσετε μια που έχετε ήδη αποθηκεύσει (αποθηκεύεται πάντα στο pal.dat, απλά πατήστε "i" κατά την επεξεργασία ενός sprite). Αν αποφασίσετε να ορίσετε τη δική σας παλέτα, θα πρέπει να επαναπρογραμματίσετε τις γραμμές BASIC όπου ορίζεται η εναλλακτική (ή "προσαρμοσμένη") παλέτα.

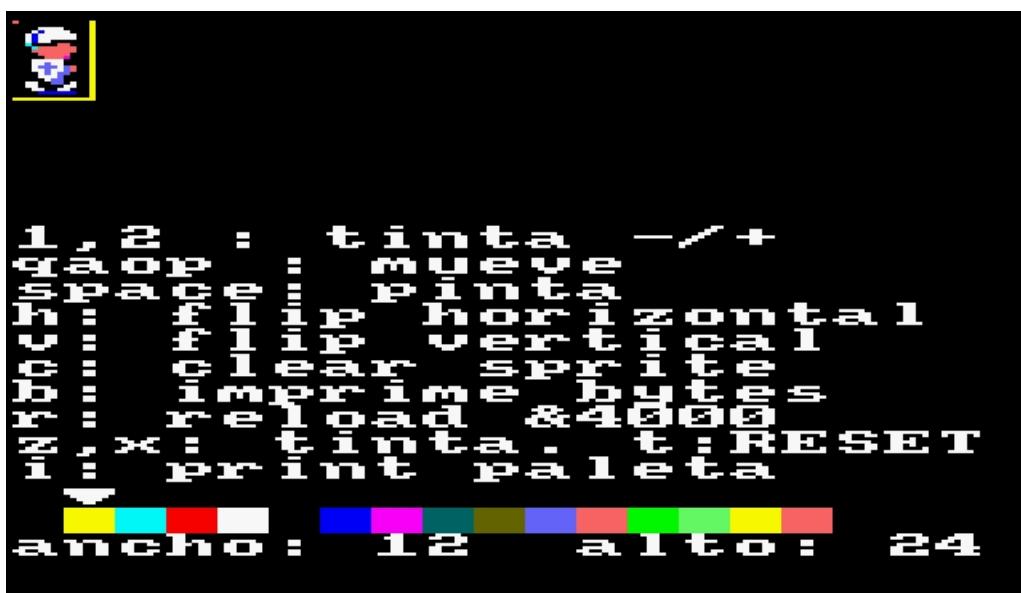
είναι μια υπορουτίνα που καλείται από το GOSUB όταν πατάτε "2" στην απάντηση στην ερώτηση για το ποια παλέτα θέλετε να χρησιμοποιήσετε.



Εικ. 21 Αρχική οθόνη SPEDIT

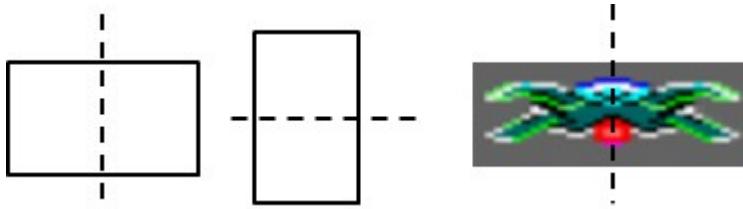
Το εργαλείο σας επιτρέπει να επιλέξετε τη λειτουργία 1 ή τη λειτουργία 0. Μόλις μπείτε στη λειτουργία επεξεργασίας, σας επιτρέπει να επεξεργαστείτε τα σχέδια, με βοήθεια στην οθόνη. Χειρίζεστε ένα pixel που αναβοσβήνει και στο κάτω μέρος εμφανίζονται οι συντεταγμένες όπου βρίσκεστε και η τιμή του byte στο οποίο βρίσκεστε.

Όταν σας ζητάει το πλάτος και το ύψος του sprite, να θυμάστε ότι το **μέγιστο ύψος ενός sprite στην 8BP είναι 127 γραμμές, καθώς και το μέγιστο πλάτος του σε bytes**. Σημειώστε επίσης ότι το πλάτος ζητείται σε pixels, αλλά θα πρέπει να γνωρίζετε ότι ο επεξεργαστής λειτουργεί εσωτερικά σε bytes, οπότε, αν πρόκειται να φτιάξετε μια εικόνα σε λειτουργία 0, το πλάτος πρέπει να είναι ζυγός αριθμός (ένα byte = 2 pixels) και αν πρόκειται να φτιάξετε μια εικόνα σε λειτουργία 1, το πλάτος πρέπει να είναι πολλαπλάσιο του 4 (ένα byte = 4 pixels).



Εικ. 22 Οθόνη επεξεργασίας SPEDIT

To SPEDIT σας επιτρέπει να "καθρεφτίσετε" την εικόνα σας για να κάνετε την ίδια φιγούρα να περπατάει προς τα αριστερά χωρίς κόπο, απλά πατώντας το H (οριζόντια αναστροφή) και το ίδιο μπορεί να γίνει και κάθετα. Σας επιτρέπει επίσης να "καθρεφτίζετε" την εικόνα σε σχέση με έναν νοητό άξονα που βρίσκεται στο κέντρο της φιγούρας, τόσο κατακόρυφα όσο και οριζόντια. Αυτό είναι πολύ χρήσιμο για συμμετρικούς ή σχεδόν συμμετρικούς χαρακτήρες, όπου ένα βοήθημα σχεδίασης είναι πάντα χρήσιμο.



Eik. 23 συμμετρικά sprites με το SPEDIT

Από την έκδοση 11 του SPEDIT, υποστηρίζεται η λειτουργία AMSTRAD 1, οπότε μπορείτε να επεξεργάζεστε sprites σε λειτουργία 1 χωρίς κανένα πρόβλημα και να χρησιμοποιείτε επίσης τον μηχανισμό καθρεφτισμού.



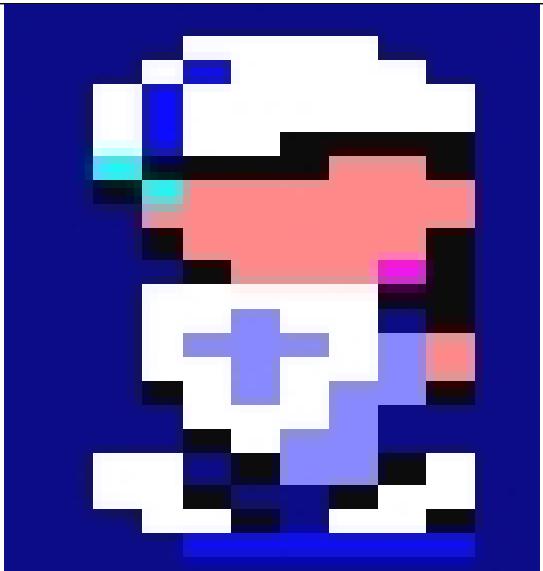
Eik. 24 επεξεργασία sprites σε MODE 1 με το SPEDIT

Αφού ορίσετε το ομοίωμά σας, για να εξαγάγετε τον κώδικα συναρμολόγησης πρέπει να πατήσετε το "b". Αυτό θα στείλει στον εκτυπωτή (στο αρχείο που έχουμε ορίσει ως έξοδο) ένα κείμενο όπως το ακόλουθο, στο οποίο μπορείτε να προσθέσετε ένα όνομα, εγώ το ονόμασα "SOLDADO_R1".

```

;----- BEGIN IMAGE -----
SOLDADO_R1
db 6 ; ancho
db 24 ; alto
db 0 , 0 , 0 , 0 , 0 , 0
db 0 , 0 , 0 , 0 , 0 , 0
db 0 , 0 , 48 , 48 , 0 , 0
db 0 , 16 , 56 , 48 , 32 , 0
db 0 , 52 , 48 , 48 , 48 , 0
db 0 , 52 , 48 , 48 , 48 , 0
db 0 , 52 , 48 , 240 , 240 , 0
db 0 , 88 , 240 , 229 , 218 , 0
db 0 , 164 , 207 , 207 , 207 , 0
db 0 , 69 , 207 , 207 , 207 , 0
db 0 , 80 , 207 , 207 , 218 , 0
db 0 , 0 , 229 , 207 , 248 , 0
db 0 , 16 , 48 , 48 , 240 , 0
db 0 , 16 , 37 , 48 , 80 , 0
db 0 , 16 , 15 , 26 , 79 , 0
db 0 , 16 , 37 , 48 , 79 , 0
db 0 , 80 , 37 , 37 , 90 , 0
db 0 , 0 , 48 , 37 , 0 , 0
db 0 , 0 , 176 , 15 , 0 , 0
db 0 , 48 , 80 , 15 , 176 , 0
db 0 , 48 , 160 , 80 , 48 , 0
db 0 , 16 , 112 , 16 , 112 , 0
db 0 , 0 , 60 , 60 , 60 , 0
db 0 , 0 , 0 , 0 , 0 , 0
;----- END IMAGE -----

```



Παρατηρήστε ότι έχω αφήσει πάντα ένα byte αριστερά στο μηδέν. Το έκανα αυτό έτσι ώστε, όταν ο στρατιώτης κινείται προς τα δεξιά, να "σβήνει τον εαυτό του", αλλιώς θα άφηνε ένα ίχνος, "μουτζουρώνοντας" την οθόνη καθώς κινείται προς τα εμπρός.

Σχ. 25 Στρατιώτης σε μορφή .asm

Μόλις φτιάξετε το πρώτο καρέ του στρατιώτη σας, μπορείτε να αφήσετε το έργο και να συνεχίσετε κάποια άλλη μέρα. Για να ζεκινήσετε από τον στρατιώτη που έχετε σχεδιάσει και να συνεχίσετε το ρετουσάρισμά του ή να το τροποποιήσετε για να φτιάξετε ένα άλλο πλαίσιο, μπορείτε να συγκεντρώσετε τον στρατιώτη στη διεύθυνση **20000**, αφαιρώντας το πλάτος και το ύψος. Μόλις συναρμολογηθεί από το winape, λέτε στο SPEDIT ότι πρόκειται να επεξεργαστείτε ένα sprite του ίδιου μεγέθους και μόλις βρεθείτε στην οθόνη επεξεργασίας πατάτε "r" (reload). Το sprite θα φορτωθεί από τη διεύθυνση **20000**, όπου δηλαδή το έχετε "συναρμολογήσει".

Ένα μεγάλο μέρος της γοητείας ενός παιχνιδιού είναι τα sprites του. Μην κάνετε τσιγκουνιές σε αυτό, κάντε το αργά και με γούστο και το παιχνίδι σας θα δείχνει πολύ καλύτερο.

```

org 20000
----- BEGIN IMAGE -----
SOLDADO_R1
;db 6 ; ancho ojo! comentamos esta linea
;db 24 ; alto ojo! comentamos esta linea
db 0 , 0 , 0 , 0 , 0 , 0
db 0 , 0 , 0 , 0 , 0 , 0
db 0 , 0 , 48 , 48 , 0 , 0
db 0 , 16 , 56 , 48 , 32 , 0
db 0 , 52 , 48 , 48 , 48 , 0
db 0 , 52 , 48 , 48 , 48 , 0
db 0 , 52 , 48 , 240 , 240 , 0
db 0 , 88 , 240 , 229 , 218 , 0
db 0 , 164 , 207 , 207 , 207 , 0
db 0 , 69 , 207 , 207 , 207 , 0
db 0 , 80 , 207 , 207 , 218 , 0
db 0 , 0 , 229 , 207 , 248 , 0
db 0 , 16 , 48 , 48 , 240 , 0
db 0 , 16 , 37 , 48 , 80 , 0
db 0 , 16 , 15 , 26 , 79 , 0
db 0 , 16 , 37 , 48 , 79 , 0
db 0 , 80 , 37 , 37 , 90 , 0
db 0 , 0 , 48 , 37 , 0 , 0
db 0 , 0 , 176 , 15 , 0 , 0
db 0 , 48 , 80 , 15 , 176 , 0
db 0 , 48 , 160 , 80 , 48 , 0
db 0 , 16 , 112 , 16 , 112 , 0
db 0 , 0 , 60 , 60 , 60 , 0
db 0 , 0 , 0 , 0 , 0 , 0
----- END IMAGE -----

```

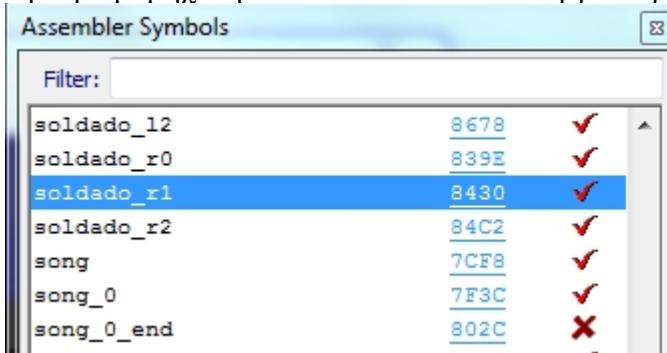
Με αυτό ξέρετε τι σημαίνει να "συναρμολογείτε" ένα sprite. Σημαίνει απλώς να τοποθετήσετε τα bytes δεδομένων που το αποτελούν σε διαδοχικές διεύθυνσεις μνήμης, στην προκειμένη περίπτωση ξεκινώντας από την **20000**.

Το SPEDIT καταλαμβάνει πολύ λίγη μνήμη και αυτή η διεύθυνση είναι μακριά από το πρόγραμμα, οπότε δεν υπάρχει κανένα πρόβλημα ότι με τη συναρμολόγησή του "καταστρέφουμε" το πρόγραμμα SPEDIT.

Σχ. 26 Συναρμολόγηση γραφικών

Για να μάθετε σε ποια διεύθυνση μνήμης έχει συγκεντρωθεί κάθε εικόνα, χρησιμοποιήστε από το μενού Winape: Assemble->symbols

Με αυτό θα δείτε μια σχέση των ετικετών που έχετε ορίσει, όπως "SOLDADO_R1" και τη διεύθυνση μνήμης (σε δεκαεξαδικό σύστημα) από την οποία έχει συναρμολογηθεί. Για να μετατρέψετε τις διεύθυνσεις από δεκαεξαδικό σε δεκαδικό μπορείτε να χρησιμοποιήσετε την αριθμομηχανή των Windows σε λειτουργία "προγραμματιστή".



Εικ. 27 λεπτομέρεια των συμβόλων που εμφανίζονται στο Winape

Αφού δημιουργήσετε τις διάφορες φάσεις κίνησης του στρατιώτη σας, μπορείτε να τις ομαδοποιήσετε σε μια "ακολουθία" κίνησης. Οι ακολουθίες κινούμενων σχεδίων είναι λίστες εικόνων και δεν ορίζονται με το SPEDIT. Με το SPEDIT απλά επεξεργάζεστε το

"frames". Σε μια επόμενη ενότητα θα εξηγήσω πώς να πείτε στη βιβλιοθήκη 8BP ποιο σύνολο εικόνων αποτελεί μια ακολουθία κινουμένων σχεδίων.

Οι εικόνες που φτιάχνετε για το παιχνίδι σας αποθηκεύονται σε ένα ενιαίο αρχείο, το οποίο ονομάζεται "images_mygame.asm", για παράδειγμα. Αυτό το αρχείο ξεκινάει με μια λίστα εικόνων στις οποίες μπορείτε να ανατρέξετε στις εντολές 8BP της BASIC με έναν δείκτη, ανεξάρτητα από τη διεύθυνση στην οποία έχουν συγκεντρωθεί, για παράδειγμα:

```
IMAGE_LIST
Η πρώτη εικόνα θα εκχωρείται πάντα στον δείκτη 16, η επόμενη
εικόνα στον δείκτη 16 και η επόμενη εικόνα στον δείκτη 16.
17 και ούτω καθεξής
-----
dw SOLDIER_R0; 16
dw SOLDIER_R1; 17
dw SOLDIER_R2; 18
```

Μόλις ολοκληρωθούν όλες οι εικόνες, μπορείτε να συγκεντρώσετε τη βιβλιοθήκη μαζί με τη μουσική και τα γραφικά.

ΠΟΛΥ ΣΗΜΑΝΤΙΚΟ: φροντίστε να μην υπερβείτε τα 8440 bytes γραφικών. Για να το κάνετε αυτό, ελέγξτε πού είναι συγκεντρωμένη η ετικέτα "END GRAPH", η οποία πρέπει να είναι μικρότερη από 42040 (αφού 42040 - 33600 = 8440 bytes). Αν συναρμολογηθεί σε υψηλότερη διεύθυνση, τότε "τσακίζετε" διευθύνσεις που χρειάζεται ο διερμηνέας BASIC και ο υπολογιστής μπορεί να καταρρεύσει. Σε περίπτωση που χρειάζεστε περισσότερη μνήμη για τα γραφικά, θα πρέπει να συναρμολογήσετε τα "επιπλέον" γραφικά σε μια ελεύθερη περιοχή μνήμης, π.χ. 22000, και να χρησιμοποιήσετε μια MEMORY 21999 στο πρόγραμμά σας, μειώνοντας τη διαθέσιμη μνήμη για τη BASIC.

8.2 Εκτύπωση ενός sprite

Ας δούμε τα βασικά στοιχεία της εκτύπωσης ενός sprite. Ας υποθέσουμε ότι έχετε σχεδιάσει έναν στρατιώτη και τον έχετε συμπεριλάβει στο αρχείο images_mygame.asm. Ας υποθέσουμε ότι είναι η πρώτη σας εικόνα και επομένως έχει το αναγνωριστικό 16.

Στην 8BP έχετε 32 sprites (αριθμημένα από το 0 έως το 31). Μπορείτε να αντιστοιχίσετε μια εικόνα σε οποιοδήποτε sprite με την εντολή |SETUPSP. Αυτή η εντολή σας επιτρέπει να τροποποιήσετε πολλά χαρακτηριστικά ενός sprite, όχι μόνο την εικόνα του. Το χαρακτηριστικό που θέλετε να τροποποιήσετε του sprite είναι η δεύτερη παράμετρος της εντολής SETUPSP.

```
|SETUPSP, <sprite id>, <parameter> , <value>, <sprite id>,
<parameter> , <value>, <value>.
```

Για να εκχωρήσετε μια εικόνα πρέπει να χρησιμοποιήσετε την παράμετρο 9. Ένα πολύ απλό πρόγραμμα που εκτυπώνει ένα sprite θα ήταν το ακόλουθο:

10 MNHMH 23499

20 CALL &6B78: REM εγκαθιστά τις εντολές RSX

30 DEFINT A-Z : REM ακέραιες αριθμητικές μεταβλητές (ταχύτερα)

40 |SETUPSP,31,9,16: REM αναθέτει την εικόνα 16 στο sprite 31

50 x=40:y=100: REM συντεταγμένες όπου θέλουμε να εκτυπώσουμε

60 |LOCATESP,31,y,x: REM τοποθετεί το sprite 31

70 |PRINTSP,31: REM εκτυπώνει το sprite 31

Αυτό θα ήταν το αποτέλεσμα



Εικ. 28 Εκτύπωση ενός sprite στην οθόνη

Θα χρησιμοποιήσετε συχνά την εντολή **SETUPSP** και σταδιακά θα τη γνωρίσετε σε βάθος. Οι παράμετροι της SETUPSP δεν είναι απλά ένας οποιοσδήποτε αριθμός. Υπάρχουν 7 πιθανές τιμές και είναι οι εξής (0,5,6,7,8,9,15):

- Παράμετρος 0: αλλαγή του byte κατάστασης του sprite
- Παράμετρος 5: αλλαγή Vy. Η Vx μπορεί επίσης να αλλάξει ταυτόχρονα αν την προσθέσουμε στο τέλος ως επιπλέον παράμετρο (όπως αυτή : SETUPSP, <id>,5, Vy,Vx)
- Παράμετρος 6: αλλαγές Vx
- Παράμετρος 7: ακολουθία αλλαγής (λαμβάνει τιμές 0..31)
- Παράμετρος 8: αλλαγή frame_id (λαμβάνει τιμές 0..7)
- Παράμετρος 9: αλλαγή εικόνας. Η καθορισμένη εικόνα μπορεί να είναι μία από την αρχική λίστα εικόνων στο αρχείο images_mygame.asm,
- Παράμετρος 15: διαδρομή αλλαγής (καταλαμβάνει 1bytes)

Καθώς θα διαβάζετε αυτό το εγχειρίδιο, θα καταλάβετε τη σημασία των χαρακτηριστικών ενός sprite και πώς να τα χρησιμοποιείτε ανάλογα με τις ανάγκες σας.

8.3 Αναδίπλωση Sprite

Σε πολλές περιπτώσεις θα χρειαστεί να σχεδιάσετε χαρακτήρες που περπατούν προς διαφορετικές κατεύθυνσεις, με διαφορετικές εικόνες για κάθε περίπτωση. Η εικόνα του sprite προς την αριστερή κατεύθυνση θα είναι η κατοπτρική εικόνα της δεξιάς κατεύθυνσης. Μπορείτε να ορίσετε δύο εικόνες και να τις αποθηκεύσετε στη μνήμη, αλλά από το V33, υπάρχει ένας τρόπος να αποφύγετε την κατανάλωση μνήμης RAM για αυτές τις εικόνες. Αυτό ονομάζεται "γυρισμένες" εικόνες.



Σχ. 29 παράδειγμα ανεστραμμένων εικόνων

Μια "αντεστραμμένη" εικόνα είναι μια κατοπτρική εικόνα μιας άλλης εικόνας που έχει δημιουργηθεί και συμπεριληφθεί στο αρχείο εικόνας. Ορίζονται μια εικόνα με αυτόν τον τρόπο, αποφεύγετε την αποθήκευση της. Για να το κάνετε αυτό, απλά συμπεριλαμβάνετε μια λίστα με τις "γυρισμένες" εικόνες στο αρχείο εικόνας (το οποίο συνήθως ονομάζω "images_mygame.asm"). Θα βρείτε στην αρχή του αρχείου ένα τμήμα που οριθετείται από τις ετικέτες "_BEGIN_FLIP_IMAGES" και "_END_FLIP_IMAGES" για το σκοπό αυτό.

```
;-----  
_BEGIN_FLIP_IMAGES  
εδώ τοποθετούνται εικόνες που ορίζονται ως άλλες υπάρχουσες  
εικόνες αλλά αναποδογυρισμένες οριζόντια.  
JOE_LEFT dw JOE_RIGHT; ο joe_left θα είναι η flip-flop έκδοση του  
joe_right  
  
Ορίζω τα καρέ του στρατιώτη στα αριστερά ως ανεστραμμένα  
SOLDIER_L0 dw SOLDIER_R0,  
SOLDIER_L1 dw SOLDIER_R1; SOLDIER_L2 dw  
SOLDIER_R2; SOLDIER_L1_UP dw  
SOLDIER_R1_UP SOLDIER_L1_DOWN dw  
SOLDIER_R1_DOWN  
  
_END_FLIP_IMAGES  
;-----
```

Οι ανεστραμμένες εικόνες μπορούν να χρησιμοποιηθούν όπως οι κανονικές εικόνες. Παρακάτω θα μάθετε πώς να δημιουργείτε ακολουθίες κινουμένων σχεδίων, τις οποίες μπορείτε να κατασκευάσετε τόσο με ανεστραμμένες όσο και με μη ανεστραμμένες εικόνες. Από κάθε άποψη, είναι σαν να είναι μια "γυρισμένη" εικόνα πραγματική, αν και πρόκειται για μια "εικονική" εικόνα, η οποία δεν αποθηκεύεται και υπολογίζεται ως καθρέφτης μιας υπάρχουσας εικόνας όταν εκτυπώνεται. Οι ανεστραμμένες εικόνες υποστηρίζονται τόσο στη λειτουργία 0 όσο και στη λειτουργία 1.

Το μειονέκτημα των ανεστραμμένων εικόνων είναι ότι η εκτύπωσή τους είναι πιο ακριβή, και συγκεκριμένα χρειάζονται 1,8 φορές περισσότερο χρόνο από μια κανονική εκτύπωση, γεγονός που μπορεί να μεταφραστεί σε χαμηλότερη ταχύτητα του παιχνιδιού σας. Εάν το παιχνίδι σας είναι ένα arcade game (ένα shoot'em up) όπου χρειάζεστε τη μέγιστη ταχύτητα, η σύστασή μου είναι να μην χρησιμοποιείτε μαζικά γυρισμένες εικόνες. Ωστόσο, σε παιχνίδια περιπέτειας, παιχνίδια με πέρασμα οθόνης, παιχνίδια λαβύρινθου κ.λπ. είναι μια εξαιρετική επιλογή. Σε κάθε περίπτωση, δοκιμάστε να τα χρησιμοποιήσετε στο arcade σας, γιατί αν δεν υπάρχουν πολλά flips ταυτόχρονα, η ταχύτητα που προκύπτει μπορεί να είναι πολύ αποδεκτή.

Έχω κάνει οριζόντια αναστροφή και όχι κάθετη αναστροφή, επειδή κανονικά ένας χαρακτήρας που περπατάει προς τα αριστερά είναι το είδωλο του ίδιου χαρακτήρα που περπατάει προς τα δεξιά, ενώ το περπάτημα προς τα πάνω δείχνει την πλάτη και το

περπάτημα προς τα κάτω δείχνει το στήθος και το πρόσωπο. Επομένως, το κάθετο flipping δεν είναι τόσο χρήσιμο όσο το οριζόντιο flipping, και προς το συμφέρον της μείωσης του μεγέθους του 8BP, δεν το συμπεριέλαβα στις δυνατότητές του.

ΣΗΜΑΝΤΙΚΟ: η αναστροφή δεν εφαρμόζεται σε εικόνες τύπου τμήματος που μπορούν να χρησιμοποιηθούν στη λειτουργία ψευδο-3D 8BP.

8.4 Sprites με αντικατάσταση

Από την έκδοση v22 της 8BP είναι δυνατή η επεξεργασία διαφανών sprites, δηλαδή sprites που μπορούν να πετάξουν πάνω από ένα φόντο και να το επαναφέρουν όταν περνούν. Για να γίνει αυτό, τα sprites που απολαμβάνουν αυτή τη δυνατότητα πρέπει να διαμορφωθούν με ένα "1" στη σημαία αντικατάστασης του status byte (bit 6). Στην επόμενη ενότητα, το status byte θα εξηγηθεί λεπτομερώς. Ας δούμε πώς μπορείτε να επεξεργαστείτε ένα sprite με αυτή τη δυνατότητα με το SPEDIT.

Πολλά παιχνίδια χρησιμοποιούν μια τεχνική που ονομάζεται "double buffering" για να μπορούν να επαναφέρουν το φόντο όταν ένα sprite μετακινείται στην οθόνη. Αυτό βασίζεται στο να έχουμε ένα αντίγραφο της οθόνης (ή της περιοχής του παιχνιδιού) σε μια άλλη περιοχή της μνήμης, έτσι ώστε ακόμα και αν τα sprites μας καταστρέψουν το φόντο, να μπορούμε πάντα να κοιτάξουμε σε αυτή την περιοχή για να δούμε τι ήταν από κάτω και να το επαναφέρουμε. Στην πραγματικότητα, αυτή είναι η βασική αρχή, αλλά είναι λίγο πιο περίπλοκο. Εκτυπώνεται στον διπλό buffer (που ονομάζεται επίσης "backbuffer") και όταν εκτυπωθεί όλο, είτε γίνεται dump στην οθόνη είτε αλλάζει η διεύθυνση έναρξης της μνήμης βίντεο από την αρχική διεύθυνση της οθόνης στη νέα, τη διεύθυνση του διπλού buffer. Η εναλλαγή είναι στιγμιαία (ανάλογα με τον τύπο του μηχανήματος). Για τη δημιουργία του επόμενου καρέ, χρησιμοποιείται η αρχική διεύθυνση οθόνης όπου η μνήμη βίντεο δεν δείχνει πλέον. Εκεί κατασκευάζεται το νέο καρέ και γίνεται επανασύνδεση, εναλλάξ, σε κάθε καρέ. Αυτές οι τεχνικές, αν και λειτουργούν πολύ καλά, έχουν μερικά μειονεκτήματα για τους σκοπούς μας: χρειάζονται περισσότερο χρόνο για την CPU και καταναλώνουν πολύ περισσότερη μνήμη (έως και 16KB επιπλέον), αφήνοντας μας πολύ λίγη μνήμη για το πρόγραμμα BASIC. Αν ένα παιχνίδι αναπτύσσεται εξ ολοκλήρου σε assembler, αυτό δεν είναι τόσο μεγάλο πρόβλημα, επειδή 10KB assembler φτάνουν πολύ μακριά, αλλά 10KB BASIC είναι πολύ λίγα. Ορισμένα βιντεοπαιχνίδια μειώνουν την περιοχή του παιχνιδιού για να μην χρησιμοποιούν τόσο πολλή μνήμη, αλλά αυτό τα κάνει λίγο φτωχότερα.

Η λύση που νιοθετήθηκε στην 8BP είναι εμπνευσμένη από τον προγραμματιστή Paul Shirley (συγγραφέα του βιβλίου "Mission Genocide"), αλλά είναι ελαφρώς διαφορετική. Θα διηγηθώ την ιστορία της 8BP άμεσα:



Εικ. 30 Sprites με αντικατάσταση σε 8BP

Η ιδέα είναι ότι **το φόντο δεν καταστρέφεται ποτέ από τα sprites που περνούν από πάνω του**, οπότε δεν χρειάζεται να το αποθηκεύσετε. Αυτή η φαινομενική "μαγεία" έχει τη λογική της: συνίσταται στην "απόκρυψη" του χρώματος του φόντου στο χρώμα του sprite που ζωγραφίζεται από πάνω του.

Στο AMSTRAD ένα εικονοστοιχείο της λειτουργίας 0 αναπαρίσταται με 4 bit, οπότε είναι δυνατόν να υπάρχουν έως και 16 διαφορετικά χρώματα από μια παλέτα 27 χρωμάτων. Έτσι, αν χρησιμοποιήσουμε ένα bit για το χρώμα φόντου και 3 bit για τα χρώματα του sprite, θα έχουμε συνολικά 2 χρώματα φόντου.

7 χρώματα + 7 χρώματα + 1 χρώμα για την ένδειξη της διαφάνειας = 9 χρώματα συνολικά. Αυτό θα μας επιτρέψει να "κρύψουμε" το χρώμα του φόντου στο χρώμα του sprite, αν και πληρώνουμε το τίμημα της μείωσης του αριθμού των χρωμάτων από 16 σε μόνο 9. Ωστόσο, ορισμένα διακοσμητικά στοιχεία στην οθόνη του παιχνιδιού μπορούν να έχουν περισσότερο χρώμα, καθώς τα sprites δεν θα περνούν από πάνω τους (όπως τα φύλλα στα δέντρα ή η οροφή στο παρακάτω παράδειγμα), οπότε μπορούμε να έχουμε μια ορισμένη ποσότητα χρώματος στο παιχνίδι μας.

Για να επεξεργαστούμε αυτό το είδος sprites πρέπει να χρησιμοποιήσουμε μια κατάλληλη παλέτα 9 χρωμάτων, όπου για κάθε χρώμα sprite χρησιμοποιούνται δύο δυαδικοί κωδικοί (που αντιστοιχούν στο 0 και στο 1 του bit του φόντου). Στο SPEDIT αν επιλέξετε την επιλογή παλέτας "2", θα έχετε μια παλέτα που ορίζεται με αυτόν τον τρόπο, αν και μπορείτε να την αλλάξετε κατά το δοκούν. Κατασκευάζεται ως εξής:

```

2300 REM ----- PALETA sprites transparentes MODE 0-----
2301 INK 0,11: REM azul claro
2302 INK 1,15: REM naranja
2303 INK 2,0 : REM negro
2304 INK 3,0:
2305 INK 4,26: REM blanco
2306 INK 5,26:
2307 INK 6,6: REM rojo
2308 INK 7,6:
2309 INK 8,18: REM verde
2310 INK 9,18:
2311 INK 10,24: REM amarillo
2312 INK 11,24 :
2313 INK 12,4: REM magenta
2314 INK 13,4 :
2315 INK 14,16 : REM naranja
2316 INK 15, 16:
2317 AMARILLO=10
2420 RETURN

```

Εικ. 31 Παράδειγμα παλέτας αντικατάστασης

Όπως μπορείτε να δείτε, μετά τα χρώματα 0 και 1, όλα τα χρώματα επαναλαμβάνονται δύο φορές. Μπορείτε να δημιουργήσετε τη δική σας παλέτα με αυτόν τον τρόπο. Μπορείτε να βοηθήσετε τον εαυτό σας συμβουλευόμενοι το παράρτημα αυτού του εγχειριδίου που είναι αφιερωμένο στην παλέτα χρωμάτων.

	<i>Paleta ejemplo</i>
000 1 =	<i>Color de fondo</i>
110 0 =	<i>Color de sprite</i>
<i>Cuando el sprite se imprime:</i>	
Fondo OR sprite = 1101 =	
<i>Cuando el sprite se marcha:</i>	
Pixel OR 0001 = 0001 =	
<i>El fondo nunca fue destruido, estaba "escondido" en el sprite</i>	

Η τεχνική θα μπορούσε να είναι για να συνοψίσω λέγοντας ότι **το φόντο δεν καταστρέφεται ποτέ στην πραγματικότητα από τα sprites**, αλλά "κρύβεται" στα ίδια τα sprites που εκτυπώνονται στο φόντο.

Σχ. 32 Μηχανισμός αντικατάστασης στην 8BP

Με τον επεξεργαστή SPEDIT μπορείτε να τροποποιείτε την παλέτα σύμφωνα με τις προτιμήσεις σας χωρίς να χρειάζεται να την επεξεργάζεστε χειροκίνητα με εντολές INK, και μπορείτε να την εξάγετε για να την αντιγράψετε στα προγράμματα BASIC μας. Η εξαγωγή γίνεται με την αποστολή των εντολών INK που συνθέτουν την παλέτα στον εκτυπωτή (ο εκτυπωτής ανακατευθύνεται σε ένα αρχείο από το winape). Έχουμε τα πλήκτρα z/x για να αλλάξουμε την παλέτα και την επιλογή "i" για να την εξάγουμε στο αρχείο **PALETA.DAT** ή να την εξάγει (είναι μια παλέτα χωρίς αντικατάσταση):

```
INK 0 , 1
INK 1 ,
INK ,
INK ,
INK , 26
INK 5 , 0
INK ,
INK , 8
INK 8 , 10
INK ,
INK 10 , 14
INK , 16
INK , 18
INK , 22
INK---- END PALETA -----
INK , 11
```

Πατώντας το πλήκτρο "i" αποθηκεύετε επίσης το αρχείο "pal.dat" στο δίσκο (το .dsk), ώστε να μπορείτε να το φορτώσετε αργότερα επιλέγοντας την επιλογή 3 για να απαντήσετε στην ερώτηση επιλογής παλέτας.

Τα sprites που χρησιμοποιείτε για την κατασκευή των σχεδίων φόντου μπορούν να έχουν μόνο τα χρώματα 0 και 1, αλλά τα sprites που χρησιμοποιείτε για τη διακόσμηση, όπου τα κινούμενα sprites δεν πρόκειται να περάσουν, μπορούν να χρησιμοποιούν και τα 9 χρώματα.

Μπορείτε επίσης να αυξήσετε τον χρωματισμό του τοπίου με στοιχεία sprite αντί για φόντο, όπως το πράσινο καζάνι στο παραπάνω παράδειγμα. Με αυτόν τον τρόπο μπορείτε να έχετε πολύ πολύχρωμα αποτελέσματα.

Το μελάνι 0001 έχει "ειδική" χρήση. Εάν επεξεργαστείτε ένα sprite που δεν χρησιμοποιεί τη σημαία αντικατάστασης, το μελάνι 1 θα είναι απλώς ένα χρώμα. Άλλα αν επεξεργαστείτε ένα sprite με ενεργή σημαία αντικατάστασης στο byte κατάστασης, κατά την εκτύπωση, αυτά τα pixel θα μείνουν άβαφα, σεβόμενα ότι βρίσκεται από κάτω. Αυτό επιτρέπει στις συγκρούσεις μεταξύ sprites να μην είναι "ορθογώνιες", αλλά να διατηρούν το σχήμα του sprite.

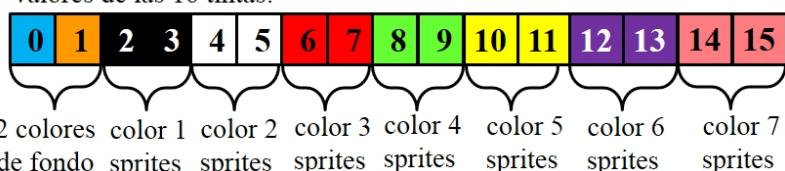
κωδικός	Σημασία	κωδικός	σημαίνει
0000	Χρώμα φόντου 1. Εάν ένα sprite το χρησιμοποιεί και ενεργοποιήσετε τη σημαία αντικατάστασης, αυτό σημαίνει "διαφάνεια", δηλαδή εκτύπωση με επαναφορά του Ταμείο	0110	χρώμα sprite 3
0001	Χρώμα φόντου 2. Εάν ένα sprite το χρησιμοποιεί και ενεργοποιήσετε τη σημαία αντικατάστασης, δεν σημαίνει πλέον χρώμα, αλλά "μην εκτυπώνετε". Ό,τι υπάρχει σε αυτό το εικονοστοιχείο γίνεται σεβαστό, π.χ. ένα εικονοστοιχείο χρωματισμένο από ένα άλλο sprite που είχε προηγουμένως εκτυπωθεί με το ότι επικαλυπτόμαστε.	0111	χρώμα sprite 4
0010		1001	χρώμα sprite 5
0011	χρώμα sprite 1	1010	χρώμα sprite 6
0100		1011	χρώμα sprite 7
0101	Στη συνέχεια, θα χρησιμοποιήσουμε αντικατάσταση sprite όπου έχω ζωγραφίσει με μελάνι 0001 ό,τι δεν πρόκειται να ζωγραφιστεί, δηλαδή, όπου το φόντο δεν πρόκειται καν να αποκατασταθεί, γιατί με τα υπόλοιπα pixel στο 0000 αρκεί να σβηστεί το ίχνος του sprite ενώ κινείται.	1101	
0110		1110	
0111		1111	

9 χρώματα συνολικά:

- 2 στο παρασκήνιο
- 7 για sprites (στην πραγματικότητα 8, αλλά ένα - 000- σημαίνει διαφάνεια)
- Το στοιχεία διακοσμητικά στοιχεία μπορούν να χρησιμοποιήσουν και τα 9.



Valores de las 16 tintas:



Eik. 33 sprite και παλέτα σχεδιασμένα για αντικατάσταση

ΣΗΜΑΝΤΙΚΟ, ΟΤΑΝ ΕΠΙΕΞΕΡΓΑΖΕΣΤΕ ΤΑ SPRITES ΣΑΣ ΜΕ OVERWRITE:

Μην χρησιμοποιείτε τα bits φόντου για τα μελάνια των sprites σας, εκτός αν θέλετε να δημιουργήσετε ειδικά εφέ, όπως περιγράφεται αργότερα σε αυτό το κεφάλαιο. Αν δεν ακολουθήσετε αυτόν τον κανόνα, μπορεί να παρατηρήσετε "περίεργα" εφέ. Δηλαδή, θα παρατηρηθούν φαινόμενα που δεν θα έχουν καμία σχέση με την πραγματικότητα:

- Αν το φόντο είναι 1 bit, τότε χρωματίστε τα sprites σας με μελάνια που τελειώνουν σε 0 (δηλαδή μελάνια 2,4,6,8,10,12,14).
- Αν το φόντο έχει 2 bit, τότε χρωματίστε τα sprites σας με μελάνια που τελειώνουν σε 00 (δηλαδή μελάνια 0100,1000,1100 που είναι τα μελάνια 4, 8 και 12 αντίστοιχα).

Όπως μπορείτε να φανταστείτε, στην περίπτωση του καζανιού, που είναι ένα sprite που δεν κινείται και επομένως δεν σβήνει τον εαυτό του, ολόκληρο το περίγραμμά του ζωγραφίζεται με το μελάνι 0001. Αυτό επιτρέπει τέλειες συγκρούσεις, χωρίς ορθογώνια σχήματα που κάνουν εμφανές ότι τα sprites είναι στην πραγματικότητα

είναι ορθογώνια. Το τελικό αποτέλεσμα είναι όπως φαίνεται παρακάτω σε μια πολλαπλή σύγκρουση.



Όπως ίσως έχετε μαντέψει, η σύγκρουση δεν είναι μόνο τέλεια, αλλά δείχνει επίσης ότι τα sprites έχουν ταξινομηθεί σύμφωνα με τη συντεταγμένη Y τους, έτσι ώστε το τελευταίο που εκτυπώνεται να είναι αυτό που βρίσκεται στη χαμηλότερη θέση. Αυτό γίνεται με μια απλή παράμετρο κατά την εκτύπωση των sprites με την εντολή |PRINTSPALL, την οποία θα δούμε αργότερα.

Σχ. 34 Πολλαπλή σύγκρουση, αποτέλεσμα μελάνης 0001

Οι λειτουργίες εκτύπωσης με αυτόν τον μηχανισμό είναι πολύ γρήγορες, χωρίς να χρειάζεται να οριστούν οι λεγόμενες "μάσκες sprite". Οι μάσκες sprite είναι bitmaps μεγέθους sprite που χρησιμεύουν για την επιτάχυνση των λειτουργιών εκτύπωσης. Σε αυτή την περίπτωση δεν είναι απαραίτητες. Το ακόλουθο σχήμα αναπαριστά μια τυπική μάσκα που σχετίζεται με ένα sprite. Πρώτα γίνεται συνήθως η πράξη AND μεταξύ του φόντου και της μάσκας και στη συνέχεια γίνεται η πράξη OR με το sprite. Στην 8BP είναι ταχύτερη, επειδή το sprite δεν αγγίζει το bit που προορίζεται για το φόντο, οπότε η πράξη OR μεταξύ του φόντου και του sprite σέβεται το φόντο ενώ ζωγραφίζει το sprite. Αν δεν το καταλαβαίνετε αυτό πολύ καλά, μην ανησυχείτε, δεν είναι σημαντικό να το καταλάβετε, καθώς δεν είναι απαραίτητο στην 8BP.

sprite	mask	Metodo convencional:
0 2 2 0	1 0 0 1	Se imprime
2 3 3 2	0 0 0 0	Fondo AND mask OR sprite
0 2 2 0	1 0 0 1	
0 2 2 0	1 0 0 1	
0 0 0 0	1 1 1 1	

Εικ. 35 Στην 8BP δεν απαιτούνται μάσκες.

Η εκτύπωση sprites με ενεργή τη σημαία αντικατάστασης είναι πιο δαπανηρή από την εκτύπωση χωρίς αντικατάσταση. Αν και δεν απαιτεί καμία μάσκα και είναι πολύ γρήγορη, χρειάζεται περίπου 1,6 φορές περισσότερο χρόνο από την εκτύπωση ενός sprite χωρίς αντικατάσταση. Για το λόγο αυτό, χρησιμοποιήστε την όταν είναι απαραίτητο και μην τη χρησιμοποιείτε αν το παιχνίδι σας δεν πρόκειται να έχει ένα σχέδιο φόντου το οποίο πρέπει να σέβονται τα sprites. Ο συνδυασμός αντικατάστασης και αναστροφής είναι ακόμη πιο δαπανηρός (χρειάζεται 2,2 φορές περισσότερο χρόνο από μια κανονική εκτύπωση χωρίς αντικατάσταση και αναστροφή), οπότε λάβετε το υπόψη στα παιχνίδια σας.

8.4.1 Χρήση αντικατάστασης για τη βελτίωση των επικαλύψεων sprite

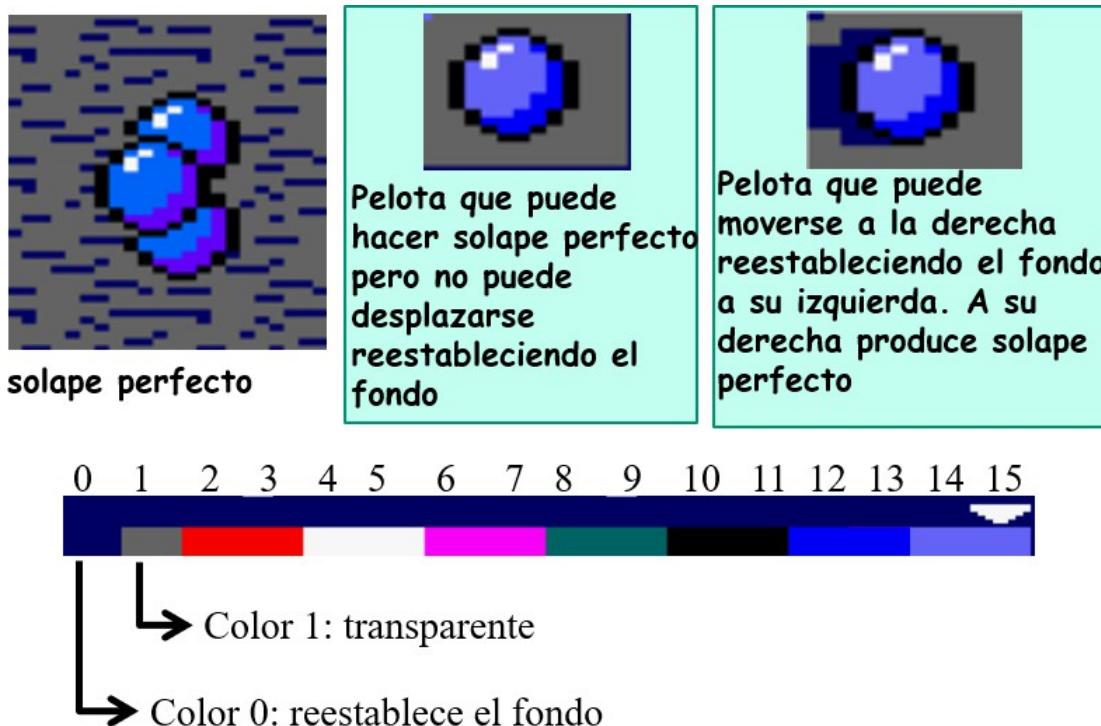
Ένα πολύ πολύτιμο χαρακτηριστικό της αντικατάστασης είναι ότι επιτρέπει στις επικαλύψεις μεταξύ των sprites να είναι "τέλειες", διότι αν όταν επεξεργάζεστε το sprite

(το οποίο πρόκειται να αντικαταστήσετε) χρησιμοποιείτε μελάνι 1 γύρω του, όταν το εκτυπώσετε όλα τα pixel που έχουν μελάνι 1 θα "αντικατασταθούν".

Θα γίνει "διαφανές", δηλαδή αν έχετε προηγουμένως ζωγραφίσει ένα άλλο sprite, τα pixel του θα γίνουν σεβαστά, με αποτέλεσμα "τέλειες" επικαλύψεις. Μπορεί να σας ενδιαφέρει αυτό το χαρακτηριστικό των sprites αντικατάστασης, ακόμα και αν το παιχνίδι σας δεν απαιτεί αντικατάσταση επειδή έχει μαύρο ή μονόχρωμο φόντο.

Θα σβηστούν μόνο τα εικονοστοιχεία που έχετε σχεδιάσει με μηδενικό μελάνι (επαναφέροντας το φόντο). Στο παράδειγμα της μπάλας, τα μηδενικά εικονοστοιχεία είναι τα πίσω εικονοστοιχεία, ώστε να μπορείτε να τα διαγράψετε όταν την μετακινήσετε προς τα δεξιά.

Με το ακόλουθο ενδεικτικό παράδειγμα μπορείτε να το καταλάβετε απόλυτα. Χάνετε χρώμα επειδή υπάρχουν μόνο 9 χρώματα, αλλά οι επικαλύψεις μεταξύ των sprites είναι πολύ καλές. Προσέξτε, γιατί χάνετε επίσης κάποια ταχύτητα εκτύπωσης.



Σχ. 36 Τέλειες επικαλύψεις

8.4.2 Αντικατάσταση με 4 χρώματα φόντου

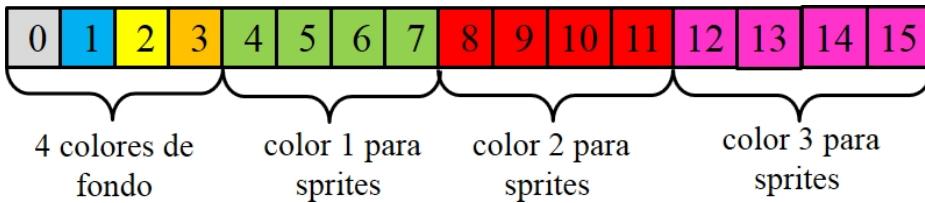
Από την έκδοση V33 είναι δυνατή η επιλογή του αριθμού των bits που χρησιμοποιούνται για το φόντο με την κλήση της εντολής |PRINTSP καθορίζοντας το Sprite 32, το οποίο δεν υπάρχει. Μπορείτε να επιλέξετε 1 ή 2 bits για το φόντο, επιτρέποντας 2 και 4 χρώματα φόντου αντίστοιχα.

|PRINTSP, 32, <αριθμός bit φόντου>.

Παραδείγματα:

PRINTSP, 32, 1 : ' με φόντο 1 bit έχουμε 2 χρώματα για το φόντο
PRINTSP, 32, 2 : ' με φόντο 2 bit έχουμε 4 χρώματα για το φόντο

Μόλις κληθεί αυτή η εντολή, η βιβλιοθήκη 8BP διαμορφώνεται ώστε να λαμβάνει υπόψη τον αριθμό των bits που θα χρησιμοποιηθούν ως bits υποβάθρου. Εάν ορίσουμε 2 bits φόντου, η παλέτα χρωμάτων μας θα πρέπει να είναι σύμφωνη με αυτή την περίσταση. Ένα παράδειγμα παρουσιάζεται παρακάτω



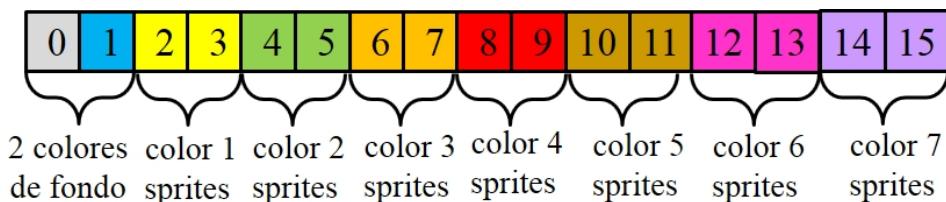
Σχ. 37 Παράδειγμα παλέτας με τέσσερα χρώματα φόντου (φόντο 2 bit)

Σε αυτό το παράδειγμα έχετε 4 χρώματα φόντου και τρία χρώματα sprite. Ακριβώς όπως όταν χρησιμοποιείτε 1 bit για το φόντο, όταν ορίζετε τα sprites σας θα πρέπει να έχετε κατά νου ότι 0 μελάνι σημαίνει διαφάνεια και 1 σημαίνει μη επαναφορά του φόντου, επιτρέποντας μη ορθογώνια σχήματα sprites. Στο ακόλουθο παράδειγμα τα 3 χρώματα που επιλέχθηκαν για τα sprites είναι το μαύρο, το ανοιχτό πράσινο και το λευκό.



Σχ. 38 παράδειγμα συνόλου παλέτας με έως και τέσσερα χρώματα φόντου (2 bit)

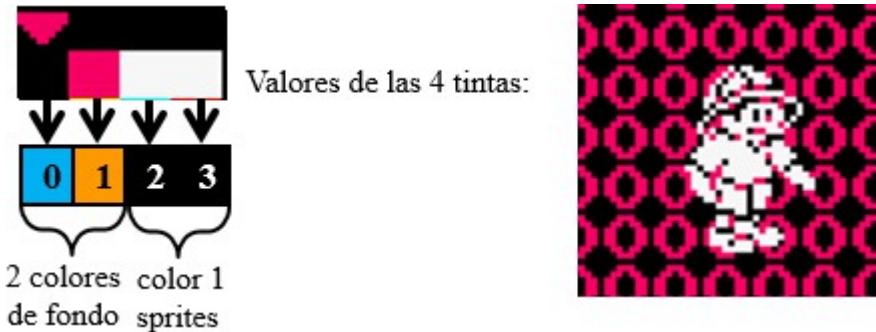
Αν και με δύο bit για το φόντο μπορείτε να έχετε πιο ωραίες διακοσμήσεις, το μειονέκτημα είναι ότι σας μένουν μόνο 3 χρώματα για τα sprites, ενώ αν χρησιμοποιήσετε 1 bit για το φόντο, έχετε μέχρι και 7 χρώματα για τα sprites σας.



Σχ. 39 Παράδειγμα παλέτας με δύο χρώματα φόντου (φόντο 1 bit)

8.4.3 Αντικατάσταση σε MODE 1

Από την έκδοση V34 της 8BP, είναι δυνατή η χρήση sprites με αντικατάσταση στη MODE 1. Εδώ έχουμε έναν πολύ ισχυρό περιορισμό επειδή, αν και έχουμε δύο χρώματα για το φόντο, έχουμε μόνο ένα χρώμα για τα sprites.

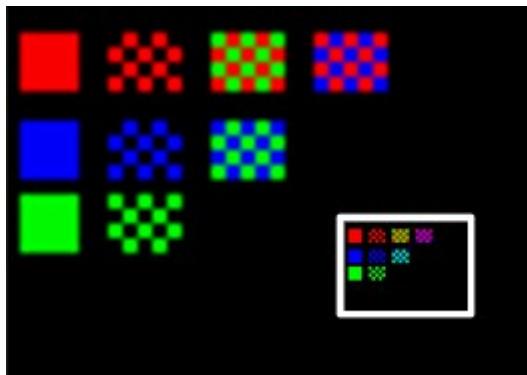


Σχ. 40 Παράδειγμα παλέτας λειτουργίας 1

Είναι εύκολο να κάνετε το λάθος να νομίζετε ότι τα sprites έχουν οποιοδήποτε χρώμα και, επιπλέον, μαύρο. Αυτό δεν ισχύει. Το μαύρο είναι ένα άλλο χρώμα και ως εκ τούτου πρέπει να καταναλώνει δύο μελάνια με αυτόν τον μηχανισμό. Έχουμε μόνο δύο μελάνια για το sprite και τα έχουμε χρησιμοποιήσει στο λευκό (σε αυτό το παράδειγμα). Όπως μπορείτε να δείτε, ο χαρακτήρας όπου δεν είναι λευκός είναι διάφανος και όχι μαύρος.

Παρά αυτόν τον αυστηρό περιορισμό, αν δουλέψετε σκληρά, μπορείτε να φτιάξετε μερικά πολύ ελκυστικά sprites στο MODE 1, και αν αλλάξετε τα χρώματα φόντου σε κάθε οθόνη και χρησιμοποιήσετε μίζεις χρωμάτων (πλέγματα) στους δείκτες του παιχνιδιού, μπορείτε να επιτύχετε ένα πολύ ικανοποιητικό αποτέλεσμα.

Χρησιμοποιώντας την ανάμειξη στη MODE 1 μπορείτε να προσομοιώσετε 10 χρώματα με μόνο 4. Εδώ μπορείτε να δείτε ένα παράδειγμα. Τα χρώματα του πλέγματος συγχωνεύονται και, για παράδειγμα, το πράσινο + κόκκινο φαίνεται κίτρινο. Μπορεί να μην το βλέπετε τόσο πειστικά σε αυτή την εικόνα, αλλά στην οθόνη του Amstrad σας θα το δείτε καλά.



8.4.4 Πώς να ζωγραφίσετε sprites "πίσω" από το φόντο

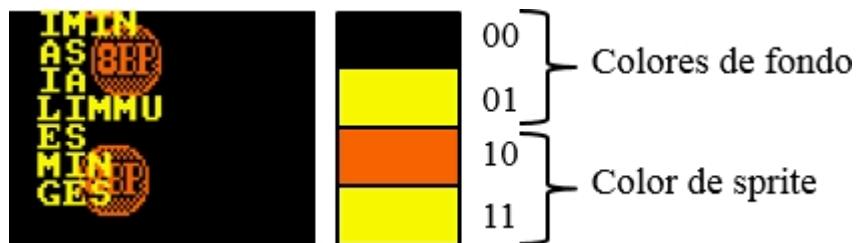
Ο ίδιος μηχανισμός για την εκτύπωση των sprites πάνω από το φόντο μπορεί να χρησιμοποιηθεί για τη ζωγραφική πίσω από το φόντο.

Όπως έχουμε ήδη δει, για να εκτυπώσουμε μπροστά από το φόντο χρησιμοποιούμε bit που δεν χρησιμοποιούνται στο φόντο, έτσι ώστε, αν και μειώνουμε τον αριθμό των χρωμάτων, να μην καταστρέφουμε τα bit του φόντου. Αν χρησιμοποιήσουμε ένα bit για το φόντο, θα πρέπει να χρησιμοποιήσουμε δύο μελάνια για να αναπαραστήσουμε το ίδιο χρώμα του sprite: ένα με το bit του φόντου στο μηδέν και ένα με το bit του φόντου στο 1.

Ωστόσο, αν αντί να αναθέσουμε το ίδιο χρώμα σε αυτά τα δύο μελάνια, αναθέσουμε το ίδιο χρώμα με το φόντο σε εκείνο με το bit φόντου στο 1, τότε αν το sprite

επικαλύπτεται με το φόντο, θα φαίνεται το χρώμα του φόντου, δίνοντας την εντύπωση ότι το sprite περνάει πίσω από αυτό. Αυτό λειτουργεί τόσο σε MODE 0 όσο και σε MODE 1.

Στο ακόλουθο παράδειγμα χρησιμοποιείται ένα bit για το φόντο, το οποίο αποτελείται από κίτρινα γράμματα σε μαύρο φόντο. Τα sprites είναι "νομίσματα" τα οποία, όπως μπορείτε να δείτε, έχουν προφανώς ζωγραφιστεί πίσω από το φόντο.



Eik. 41 Sprites τυπωμένα "πίσω" από τα γράμματα

8.4.5 Πώς να χρησιμοποιήσετε περισσότερα χρώματα με αντικατάσταση

Αν έχετε κάνει τις πρώτες σας δοκιμές με την αντικατάσταση και χρειάζεστε περισσότερα χρώματα στο παιχνίδι σας, υπάρχουν τρεις τρόποι για να το πετύχετε, αλλά πρέπει να κατανοήσετε τη μέθοδο 8BP:

- 1) Χρησιμοποιήστε μερικά sprites με αντικατάσταση και μερικά χωρίς αντικατάσταση
- 2) Χρησιμοποιήστε sprites των οποίων το χρώμα εξαρτάται από το φόντο (χρώμα υπό συνθήκη).
- 3) Χρήση ενός sprite ως φόντο

Ας δούμε αυτά τα τρία "κόλπα" ένα προς ένα:

Χρησιμοποιήστε μερικά sprites με αντικατάσταση και μερικά χωρίς:

Αυτό είναι το απλούστερο και πιο συχνά χρησιμοποιούμενο από τα τρία κόλπα. Ας υποθέσουμε ότι μόνο ένα από τα sprites σας απαιτεί αντικατάσταση και καταναλώνει 3 χρώματα. Αυτό σημαίνει ότι πρέπει να διαθέσετε 6 μελάνια σε αυτό το sprite. Ωστόσο, εάν τα υπόλοιπα sprites δεν απαιτούν αντικατάσταση, μπορείτε να τα εκτυπώσετε χωρίς αντικατάσταση και να χρησιμοποιήσετε περισσότερα χρώματα σε αυτά, δηλ.:

- 2 μελάνια για το φόντο (2 χρώματα)
- 6 μελάνια για το Sprite με αντικατάσταση (3 χρώματα)
- 8 μελάνια για τα άλλα sprites (8 χρώματα)

Σε αυτή την περίπτωση, μπορείτε να χρησιμοποιήσετε συνολικά $2+3+8 = 13$ χρώματα!!!!. Απλά πρέπει να ενεργοποιήσετε τη σημαία αντικατάστασης στο sprite που τη χρειάζεται και να την αφήσετε ανενεργή στα υπόλοιπα sprites. Στα sprites που δεν χρησιμοποιούν overwrite μπορείτε να χρησιμοποιήσετε και τα 13 χρώματα, στο sprite με overwrite θα χρησιμοποιήσετε 3 και στο φόντο θα χρησιμοποιήσετε 2.

Είναι δυνατόν να υπάρχουν και άλλα παραδείγματα. Για παράδειγμα, αν τα sprites με αντικατάσταση χρειάζονται 4 χρώματα, τότε θα χρησιμοποιούν 8 μελάνια. Επιπλέον θα έχουμε 2 μελάνια για το φόντο και τα υπόλοιπα 6 μελάνια μπορούν να προσδιορίσουν 6 διαφορετικά χρώματα, δηλαδή μπορούμε να χρησιμοποιήσουμε συνολικά 12 χρώματα.

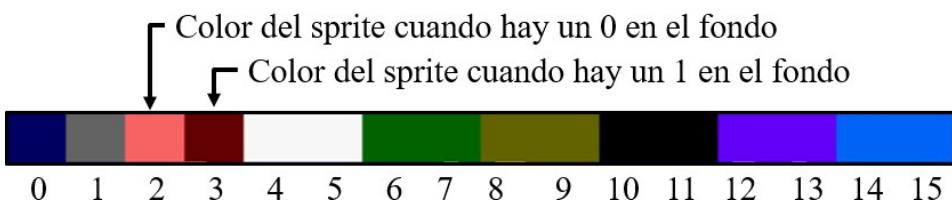
Χρησιμοποιήστε sprites των οποίων το χρώμα εξαρτάται από το φόντο (χρώμα υπό συνθήκη):

Συνίσταται στη χρήση μιας παλέτας όπου αντί να επαναλαμβάνουμε κάθε χρώμα, χρησιμοποιούμε δύο διαφορετικά χρώματα. Στην περίπτωση αυτή, όταν το φόντο είναι μηδέν θα έχουμε ένα sprite ενός χρώματος και όταν το φόντο είναι 1 θα έχουμε ένα άλλο χρώμα. Αυτό μπορεί να είναι χρήσιμο για να σχεδιάσετε λευκά πουλιά που πετούν πάνω από έναν μπλε ουρανό (χρώμα 0) ενώ κόκκινες αρκούδες περπατούν σε καφέ έδαφος (χρώμα 1). Με άλλα λόγια, μπορούμε να χρησιμοποιήσουμε περισσότερα χρώματα αρκεί η υφή του ουρανού ή του εδάφους

Η αρκούδα δεν θα έχει πάρα πολλές αλλαγές χρώματος, επειδή κάθε φορά που η αρκούδα περνάει πάνω από ένα μπλε εικονοστοιχείο φόντου θα βλέπουμε ένα λευκό εικονοστοιχείο, και αν ένα πουλί περνάει πάνω από ένα καφέ εικονοστοιχείο φόντου θα βλέπουμε ένα κόκκινο εικονοστοιχείο. Ας δούμε ένα παράδειγμα:

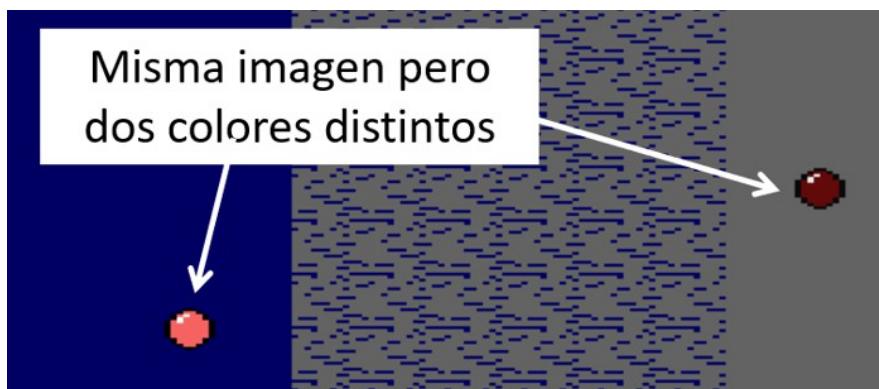


El brillo blanco siempre será blanco pero el color de la bola depende del color del fondo pues se ha definido una paleta con dos colores: uno para cuando en el fondo hay un 0 y otro para cuando hay un 1



Εικ. 42 Sprites που δημιουργήθηκαν με χρώμα "υπό όρους"

Αφού δημιουργηθεί το sprite με χρώμα υπό συνθήκη, στην παρακάτω εικόνα μπορούμε να δούμε το αποτέλεσμα που έχει όταν εκτυπώνεται σε δύο περιοχές της οθόνης, η μία με φόντο 0 και η άλλη με φόντο 1.



Εικ. 43 Χρωματικό εφέ "Conditional"

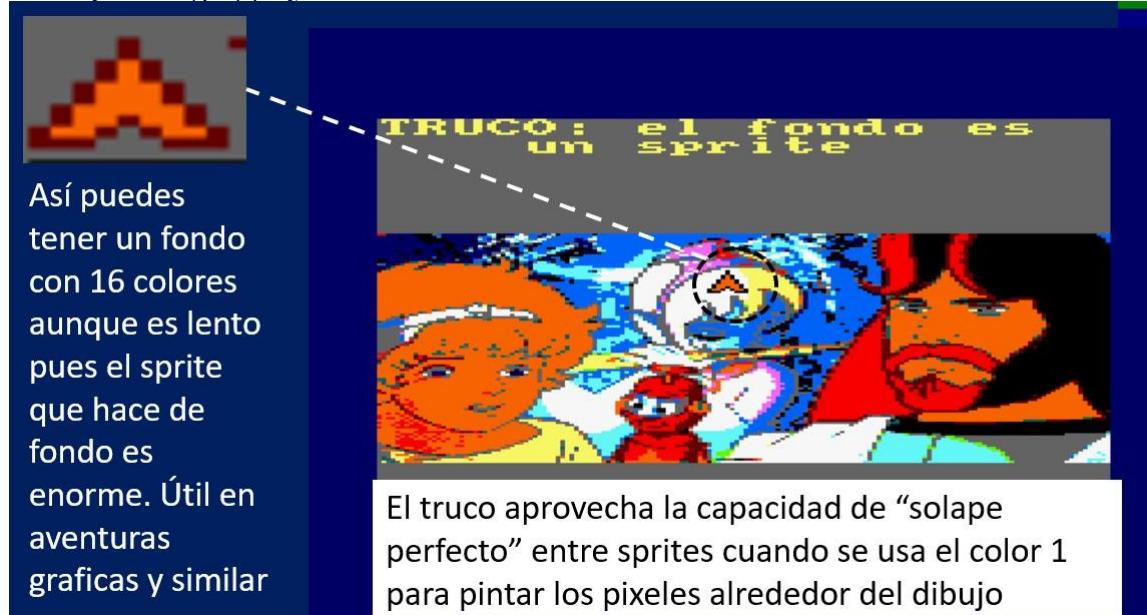
Χρησιμοποιήστε ένα sprite ως φόντο:

Αυτή η απάτη σας επιτρέπει να χρησιμοποιήσετε έως και 16 χρώματα φόντου με αντικατάσταση. Το τέχνασμα βασίζεται στο γεγονός ότι, όταν ενεργοποιείται η αντικατάσταση σε ένα sprite, όλα τα pixel που ορίζονται με "1" θα σέβονται την τιμή του pixel στην οθόνη, είτε πρόκειται για pixel από πραγματικό φόντο είτε από ένα προηγουμένως εκτυπωμένο sprite. Θα εκτυπώνουμε σε κάθε κύκλο τόσο το sprite που είναι το φόντο όσο και τα sprites που θα υπερτυπωθούν. Η τουλάχιστον σε κάθε κύκλο που τα μετακινούμε.

Το sprite που θα χρησιμοποιήσουμε ως φόντο, θα το εκτυπώσουμε χωρίς να ενεργοποιήσουμε την αντικατάσταση, ενώ τα sprites που θα εκτυπώσουμε από πάνω του θα την έχουν ενεργοποιήσει. Το μόνο μειονέκτημα είναι ότι αν το sprite του φόντου είναι πολύ μεγάλο, θα πάρει πολύ χρόνο για να εκτυπωθεί και ο αριθμός των fps του παιχνιδιού δεν θα επιτρέψει να φτιάξετε ένα arcade game, αν και μπορεί να είναι χρήσιμο για παιχνίδια περιπέτειας. Επίσης, να θυμάστε ότι στην 8BP το μέγιστο ύψος

ενός sprite είναι 127 γραμμές. Το μέγιστο πλάτος δεν αποτελεί πρόβλημα, καθώς είναι επίσης 127 bytes και η οθόνη έχει πλάτος μόνο 80 bytes.

Θα δημιουργήσουμε το sprite που θα εκτυπώσουμε στην κορυφή περιτριγυρισμένο από μονάδες, χωρίς μηδενικά, καθώς δεν θα μηδενίζει το φόντο όταν κινείται. Κάθε κύκλος θα εκτυπώνει τόσο το sprite φόντου όσο και τα μικρά sprites μας από πάνω του. Σε αυτό το παράδειγμα έχουμε σχεδιάσει ένα είδος δείκτη ή βέλους που ελέγχετε με το πληκτρολόγιο, σε ένα φόντο που καταλαμβάνει τη μισή οθόνη, δηλαδή 8KB (80bytes πλάτος x 100 γραμμές).



Εικ. 44 Ένα sprite ως φόντο

Καθώς το sprite του δείκτη έχει αντικατάσταση, θα υποστεί τα αποτελέσματα του προηγούμενου κόλπου (χρώμα υπό συνθήκη), εκτός αν ορίσετε κάποια "διπλά" χρώματα για να μην υποστεί αυτό το αποτέλεσμα. Δηλαδή, αν για παράδειγμα επαναλάβετε ένα χρώμα ώστε να μην υποστεί αυτό το φαινόμενο σε ένα sprite, τότε θα έχετε μια παλέτα με 15 διαφορετικά χρώματα και όχι 16. Δηλαδή, με κάποιο τρόπο θα πρέπει να εφαρμόσετε την αρχή του πρώτου κόλπου: ένα sprite χωρίς overwrite ως φόντο και ένα ή περισσότερα sprites με overwrite που εκτυπώνονται πάνω από αυτό. Όσο περισσότερο χρώμα έχει το φόντο, τόσο λιγότερο χρώμα θα έχουν τα sprites σας με επικάλυψη. Για παράδειγμα (είναι δυνατοί και άλλοι συνδυασμοί) μπορείτε να χρησιμοποιήσετε:

- **Φόντο:** 2 χρώματα φόντου + 8 χρώματα + 3 επαναλαμβανόμενα χρώματα = 13 χρώματα
- **Υπερεκτυπωμένα sprites:** 6 μελάνια = 3 επαναλαμβανόμενα χρώματα

Ένας τρόπος για να επιταχύνετε την ταχύτητα του "γιγαντιαίου" sprite φόντου είναι να το σπάσετε σε λωρίδες. Για παράδειγμα, σε 8 οριζόντια sprites ύψους 16. Τα οριζόντια τεντωμένα sprites εκτυπώνονται ταχύτερα από τα κάθετα τεντωμένα sprites. Σε αυτή την περίπτωση χρειάζεται να εκτυπώσετε μόνο τη λωρίδα ή το ζεύγος λωρίδων που τέμνει το sprite δείκτη σας.

8.5 Sprites με εικόνες φόντου

Οι εικόνες φόντου αποτελούν χαρακτηριστικό της 8BP V42. Η ιδέα είναι ότι σε μια κύλιση, δέντρα ή σπίτια μπορούν να περνούν κάτω από το αεροπλάνο σας και να μην προκαλούν τρεμοπαίξιμο του αεροπλάνου. Ακόμη και αν το αεροπλάνο έχει διαφανή

εκτύπωση και είναι ζωγραφισμένο στο φόντο, αν το φόντο κινείται, δεν θα σέβεται το sprite και θα προκαλεί τρεμόπαιγμα. Με αυτή τη νέα δυνατότητα, μπορείτε να φτιάξετε παιχνίδια με κύλιση όπου ο πρωταγωνιστής ή το σκάφος σας περνάει πάνω από πράγματα και **δεν υπάρχει τρεμόπαιγμα**. Για να το καταλάβετε αυτό καλύτερα, πρέπει να κατανοήσετε τον μηχανισμό κύλισης της 8BP, ο οποίος εξηγείται παρακάτω.

Στο σετ επίδειξης 8BP περιλαμβάνεται ένα που δείχνει το αποτέλεσμα της χρήσης εικόνων "φόντου" στην κύλιση.



Εικόνες φόντου σπιτιών

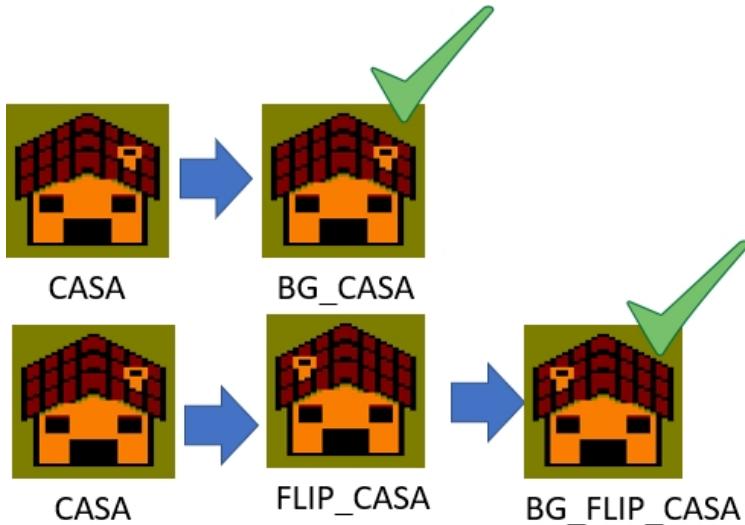
Χωρίς αυτή τη δυνατότητα, (πριν από την έκδοση V42 της 8BP) τα sprites μπορούσαν να αντικατασταθούν χωρίς τρεμοπαίζιμο εφόσον το φόντο δεν μετακινούνταν, αλλά η κίνηση του φόντου (η κύλιση) προκαλούσε αναπόφευκτο τρεμοπαίζιμο. Από την έκδοση V42 είναι δυνατή η δημιουργία εικόνων φόντου και η ανάθεσή τους σε sprites.

Αυτό που πρέπει να κάνετε είναι απλά να αναθέσετε στο Sprite μια εικόνα φόντου, η οποία στο αρχείο εικόνας πρέπει να περιέχεται μέσα στις ετικέτες:

```
_BEGIN_BG_IMAGES  
BG_HOME DW  
HOME  
BG_HOME_FLIP DW HOUSE_FLIP  
_END_BG_IMAGES
```

Η εικόνα **CASA** είναι μια εικόνα που συνήθως δημιουργείται με μια ιδιαιτερότητα: χρησιμοποιεί μόνο τα χρώματα φόντου. Έτσι, όταν ένα Sprite σε ένα παιχνίδι έχει την εικόνα **BG_CASA** που του έχει ανατεθεί, θα του έχει αυτόματα ανατεθεί ένας ειδικός τύπος διαφάνειας, στον οποίο μόνο τα bits που αντιπροσωπεύουν τα χρώματα του φόντου θα τροποποιηθούν και κάθε Sprite πάνω από το σχέδιο θα γίνει σεβαστό, αποφεύγοντας το τρεμόπαιγμα.

Η διαδικασία δημιουργίας εικόνων φόντου είναι πολύ απλή, αλλά και πολύ αυστηρή: Από μια εικόνα (χρησιμοποιώντας μόνο τα μελάνια φόντου) μπορείτε να δημιουργήσετε μια εικόνα φόντου. Εάν θέλετε να χρησιμοποιήσετε μια ανεστραμμένη εικόνα, πρέπει πρώτα να αναστρέψετε την εικόνα και στη συνέχεια να δημιουργήσετε την εικόνα φόντου με την ανεστραμμένη εικόνα.



Σε περίπτωση που θέλετε να αναποδογυρίσετε μια εικόνα φόντου, πρέπει πρώτα να την αναποδογυρίσετε με την ενότητα **FLIP_IMAGES** και στη συνέχεια να τη χρησιμοποιήσετε για να δημιουργήσετε μια εικόνα φόντου. Αυτή πρέπει να είναι η σειρά και όχι το αντίστροφο. Δηλαδή, δεν μπορείτε να δημιουργήσετε μια εικόνα, να δημιουργήσετε μια εικόνα φόντου με αυτήν και στη συνέχεια να προσπαθήσετε να αναποδογυρίσετε μια εικόνα φόντου.

Οι εικόνες φόντου κάθε φορά που εκχωρούνται σε ένα Sprite εκτυπώνονται με αυτή την ειδική διαφάνεια και δεν έχει σημασία αν το Sprite έχει εκχωρήσει ή όχι τη σημαία διαφάνειας στο byte κατάστασης (θα δούμε τώρα τι είναι αυτό).

ΣΗΜΑΝΤΙΚΟ: οι εικόνες φόντου είναι ακριβές στην εκτύπωση, και αν τις γυρίσετε, είναι ακόμη πιο ακριβές. Αν το παιχνίδι σας έχει κύλιση, προσπαθήστε να τις χρησιμοποιήσετε για στοιχεία πάνω από τα οποία θα πετάει ο χαρακτήρας ή οι εχθροί σας. Για παράδειγμα, για να επιταχύνετε την κύλιση, χρησιμοποιήστε κανονικές εικόνες σε σπίτια ή βράχους που εμφανίζονται στις πλευρές της κύλισης, οι οποίες δεν θα επικαλύπτονται συχνά από sprites.

8.6 Πίνακας χαρακτηριστικών Sprite

Τα sprites αποθηκεύονται σε έναν πίνακα που περιέχει συνολικά 32 sprites.

Κάθε εγγραφή στον πίνακα περιέχει όλα τα χαρακτηριστικά του sprite και καταλαμβάνει 16 bytes για λόγους απόδοσης, καθώς το 16 είναι πολλαπλάσιο του 2 και αυτό επιτρέπει την πρόσβαση σε οποιοδήποτε sprite με έναν πολύ φθηνό πολλαπλασιασμό. Ο πίνακας βρίσκεται στη διεύθυνση μνήμης 27000, οπότε μπορεί να γίνει προσπέλαση από τη BASIC με τις εντολές PEEK και POKE, αλλά υπάρχουν επίσης διαθέσιμες εντολές RSX για τον χειρισμό των δεδομένων αυτού του πίνακα, όπως οι εντολές |SETUPSP ή |LOCATESP.

Τα sprites έχουν ένα σύνολο παραμέτρων, η πρώτη από τις οποίες είναι το byte status flags. Σε αυτό το byte, κάθε bit αντιπροσωπεύει μια σημαία και κάθε σημαία σημαίνει ένα πράγμα, δηλαδή αν το sprite λαμβάνεται υπόψη κατά την εκτέλεση ορισμένων λειτουργιών. Ο παρακάτω πίνακας συνοψίζει τι συμβαίνει αν είναι ενεργές (στο "1")

7 ROUTEALL lo ruta	6 Sobre- escriitura	5 COLSPALL collider	4 MOVERALL lo mueve	3 AUTOALL lo mueve	2 ANIMALL lo anima	1 COLSP collided	0 PRINTSPALL lo imprime
--------------------------	---------------------------	---------------------------	---------------------------	--------------------------	--------------------------	------------------------	-------------------------------

Σχ. 45 σημαίες στο byte κατάστασης

Για να καταλάβετε τη δύναμη αυτών των σημαιών, ας δούμε μερικά παραδείγματα:

- **Bit 0:** σημαία εκτύπωσης: ο χαρακτήρας μας ή τα εχθρικά πλοία θα την έχουν ενεργοποιήσει και σε κάθε κύκλο παιχνιδιού θα καλούμε |PRINTSPALL και θα εκτυπώνονται όλα ταυτόχρονα.
- **Bit 1:** σημαία σύγκρουσης: ένα φρούτο ή ένα νόμισμα, για παράδειγμα, μπορεί να μην έχει σημαία εκτύπωσης, αλλά μπορεί να έχει σημαία σύγκρουσης. Αυτή η σημαία σημαίνει ότι ένα "συγκρουόμενο" sprite μπορεί να συγκρουστεί με το sprite που έχει ενεργή αυτή τη σημαία.
- **Bit 2:** σημαία αυτόματης κίνησης: λαμβάνεται υπόψη στο |ANIMALL. Στην περίπτωση του χαρακτήρα, προτείνω την απενεργοποίησή του, διότι αν μείνω ακίνητος, δεν χρειάζεται να αλλάξω το καρέ.
- **Bit 3:** Σημαία αυτόματης κίνησης. Κινείται μόνο όταν καλείται το AUTOALL, λαμβάνοντας υπόψη την ταχύτητά του. Χρήσιμο σε μετεωρίτες και φρουρούς που έρχονται και φεύγουν.
- **Bit 4:** σημαία σχετικής μετακίνησης. Όλα τα sprites με αυτή τη σημαία κινούνται ταυτόχρονα όταν καλείται η εντολή "|MOVEALL, dy, dx", πολύ χρήσιμη σε σχηματισμούς πλοίων και αφίξεις πλανητών. Μπορεί επίσης να χρησιμοποιηθεί για την προσομοίωση μιας κύλισης, αν αφήσετε τον χαρακτήρα σας στο κέντρο και πατώντας τα χειριστήρια κυλήσει σπίτια ή τα γύρω στοιχεία. Θα φαίνεται σαν ο χαρακτήρας σας να κινείται μέσα σε μια περιοχή.
- **Bit 5:** σημαία συγκρουστήρα. Όλα τα sprites με αυτή τη σημαία ενεργή λαμβάνονται υπόψη από τη συνάρτηση |COLSPALL, όταν ανιχνεύεται η πιθανή σύγκρουσή τους με τα υπόλοιπα sprites.
- **Bit 6:** σημαία αντικατάστασης: αν αυτή η σημαία είναι ενεργή, το sprite θα μπορεί να κινηθεί πάνω από ένα φόντο, μηδενίζοντάς το κατά το πέρασμα. Αυτή είναι μια προηγμένη επιλογή και περιλαμβάνει τη χρήση μιας ειδικά προετοιμασμένης παλέτας χρωμάτων, πιο περιορισμένης σε αριθμό χρωμάτων. Η αντικατάσταση έρχεται με αυτό το "τίμημα".
- **Bit 7:** σημαία διαδρομής: αν αυτή η σημαία είναι ενεργοποιημένη, η εντολή |ROUTEALL θα σας επιτρέψει να μετακινήσετε ένα sprite κυκλικά μέσα από μια διαδρομή που ορίζετε εσείς, η οποία ορίζεται από μια σειρά τμημάτων. Η εντολή |ROUTEALL γνωρίζει σε ποιο τμήμα και σε ποια θέση βρίσκεται κάθε sprite και αν φτάσει σε αλλαγή τμήματος, τροποποιεί την ταχύτητα του sprite σύμφωνα με τις συνθήκες του επόμενου τμήματος. |ROUTEALL δεν μετακινεί το sprite, αλλά μόνο τροποποιεί την ταχύτητα του. Για να το μετακινήσετε, πρέπει να χρησιμοποιηθεί σε συνδυασμό με το |AUTOALL.

Παραδείγματα ανάθεσης τιμών byte κατάστασης:

Τυπικός εχθρός: ένα sprite που εκτυπώνεται κάθε κύκλο, με ανίχνευση σύγκρουσης με άλλα sprites και animation πρέπει να έχει:

$$\text{κατάσταση} = 1 \text{ (bit 0)} + 2 \text{ (bit1)} + 4 \text{ (bit 2)} = 7 = \&x0111$$

Ένα σπίτι που κινείται καθώς κινούμαστε: ένα sprite που εκτυπώνεται σε κάθε κύκλο, αλλά χωρίς ανίχνευση σύγκρουσης και σχετική κίνηση.

status = 1(bit 0) + 0 (bit1) + 0 (bit 2) + 0 (bit 3) + 16 (bit 4)=17 =&x10001

Ένα φρούτο μπόνους: είναι ένα sprite που δεν εκτυπώνεται σε κάθε κύκλο, αλλά έχει ανίχνευση σύγκρουσης.

$$\text{κατάσταση} = 0 \text{ (bit 0)} + 2 \text{ (bit1)} = 2 = \&x10$$

Ένα πλοίο που θα ακολουθήσει μια προκαθορισμένη τροχιά. Θα χρειαστεί τη σημαία διαδρομής, τη σημαία αυτόματης κίνησης, τη σημαία κίνησης, τη σημαία σύγκρουσης και τη σημαία εκτύπωσης. Αυτή τη φορά θα το βάλω απευθείας σε δυαδική μορφή. Όπως μπορείτε να δείτε έχω θέσει το bit 7 σε 1, στη συνέχεια υπάρχουν 3 μηδενικά επειδή δεν έχω θέσει τις σημαίες overwrite, collider και relative movement και τέλος έχω θέσει 4 ενεργές σημαίες που αντιστοιχούν αντίστοιχα στις σημαίες automatic movement, animation, collision και printing.

status=10001111

Ο πίνακας χαρακτηριστικών sprite αποτελείται από 32 καταχωρήσεις των 16 bytes η καθεμία, ξεκινώντας από τη διεύθυνση 27000.

Ο λόγος για την ύπαρξη 16 bytes δεν είναι άλλος από την απόδοση, αφού ο υπολογισμός της διεύθυνσης του sprite Ν περιλαμβάνει πολλαπλασιασμό επί 16, ο οποίος, ως πολλαπλάσιο του 2, μπορεί να γίνει με μια μετατόπιση. Αυτό είναι χρήσιμο για λειτουργίες που αφορούν ένα μόνο sprite. Για λειτουργίες που διατρέχουν τον πίνακα sprite (όπως |PRINTSPALL ή |COLSP), εσωτερικά ο πίνακας διατρέχεται με έναν δείκτη στον οποίο προστίθεται το 16 για να μετακινηθείτε από το ένα sprite στο επόμενο. Η πρόσθεση είναι η ταχύτερη σε αυτή την περίπτωση.

Τα χαρακτηριστικά που έχει κάθε sprite είναι:

χαρακτηριστικό	Byte	Μήκος (bytes)	Σημασία
κατάσταση	0	1	Byte που περιέχει τις σημαίες κατάστασης για τις λειτουργίες PRINTSPALL, COLSPALL, ANIMALL, AUTOALL, MOVERALL, COLSPALL και ROUTEALL.
Y	1		Συντεταγμένη Y [-32768..32768] το οι τιμές που αντιστοιχούν στο εσωτερικό της οθόνης είναι [0..199].
X			Συντεταγμένη X σε bytes[-32768..32768] οιαντίστοιχες τιμές στο εσωτερικό της οθόνης είναι [0..79].
Vy	5	1	Βήμα προς την αυτόματη κίνηση
Vx		1	Βήμα προς την αυτόματη κίνηση
Ακολουθία		1	Αναγνωριστικό ακολουθίας κίνησης [0..31]. Εάν δεν έχει ακολουθία, πρέπει να εκχωρηθεί ένα μηδέν.
Ακόμα από	8	1	Αριθμός πλαισίου στην ακολουθία [0..7].
Εικόνα			Διεύθυνση μνήμης όπου βρίσκεται η εικόνα
Προηγούμενο Sprite	10		Εσωτερική χρήση για μηχανισμό ταξινόμησης sprite

Sprite επόμενο			Εσωτερική χρήση για μηχανισμό ταξινόμησης sprite
Διαδρομή		1	Αναγνωριστικό διαδρομής που πρέπει να ακολουθήσει το sprite

Η διεύθυνση μνήμης όπου αποθηκεύονται οι συντεταγμένες κάθε sprite μπορεί να υπολογιστεί ως εξής:

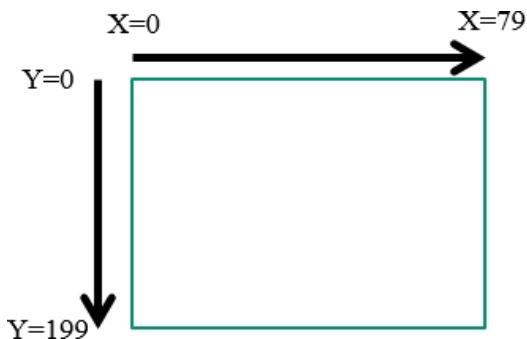
$$\Delta \text{Ιεύθυνση συντεταγμένων } Y = 27000 + 16 * N$$

$$+1 \Delta \text{Ιεύθυνση συντεταγμένων } X = 27000 +$$

$$16 * N + 3$$

Με πρόσβαση με **POKE** σε αυτές τις διευθύνσεις μπορούμε να τροποποιήσουμε την τιμή τους, αν και μπορείτε επίσης να χρησιμοποιήσετε το **|LOCATESP**.

Η βιβλιοθήκη 8BP δεν χρησιμοποιεί "εικονοστοιχεία" στη συντεταγμένη X, αλλά bytes, οπότε η συντεταγμένη X που εμπίπτει στο εσωτερικό της οθόνης βρίσκεται στο εύρος [0..79]. Η συντεταγμένη Y αναπαρίσταται σε γραμμές, οπότε το αναπαραγώγιμο εύρος στην οθόνη είναι [0..200]. Εάν τοποθετήσετε ένα sprite εκτός αυτών των περιοχών, αλλά μέρος του sprite βρίσκεται στην οθόνη, η βιβλιοθήκη θα κάνει την αποκοπή και θα ζωγραφίσει το μέρος που πρέπει να φαίνεται.



Eik. 46 Συντεταγμένες οθόνης

Οι διευθύνσεις των χαρακτηριστικών των 32 sprites μπορούν να αντιμετωπιστούν με PEEK και POKE, αν και η ακολουθία κίνησης και η ανάθεση διαδρομής περιλαμβάνουν περισσότερες λειτουργίες και αν θέλετε να τις αλλάξετε, δεν αρκεί ένα POKE, αλλά πρέπει να χρησιμοποιήσετε το **|SETUPSP**. Ακολουθεί ο κατάλογος διευθύνσεων όλων των χαρακτηριστικών των 32 sprites:

	1byte	2 bytes	2 bytes	1byte	1byte	1byte	1byte	2 bytes	1byte
sprite	status	coordy	coordx	vy	vx	seq	frame	Imagen	ruta
0	27000	27001	27003	27005	27006	27007	27008	27009	27015
1	27016	27017	27019	27021	27022	27023	27024	27025	27031
2	27032	27033	27035	27037	27038	27039	27040	27041	27047
3	27048	27049	27051	27053	27054	27055	27056	27057	27063
4	27064	27065	27067	27069	27070	27071	27072	27073	27079
5	27080	27081	27083	27085	27086	27087	27088	27089	27095
6	27096	27097	27099	27101	27102	27103	27104	27105	27111
7	27112	27113	27115	27117	27118	27119	27120	27121	27127
8	27128	27129	27131	27133	27134	27135	27136	27137	27143
9	27144	27145	27147	27149	27150	27151	27152	27153	27159
10	27160	27161	27163	27165	27166	27167	27168	27169	27175
11	27176	27177	27179	27181	27182	27183	27184	27185	27191
12	27192	27193	27195	27197	27198	27199	27200	27201	27207
13	27208	27209	27211	27213	27214	27215	27216	27217	27223
14	27224	27225	27227	27229	27230	27231	27232	27233	27239
15	27240	27241	27243	27245	27246	27247	27248	27249	27255
16	27256	27257	27259	27261	27262	27263	27264	27265	27271
17	27272	27273	27275	27277	27278	27279	27280	27281	27287
18	27288	27289	27291	27293	27294	27295	27296	27297	27303
19	27304	27305	27307	27309	27310	27311	27312	27313	27319
20	27320	27321	27323	27325	27326	27327	27328	27329	27335
21	27336	27337	27339	27341	27342	27343	27344	27345	27351
22	27352	27353	27355	27357	27358	27359	27360	27361	27367
23	27368	27369	27371	27373	27374	27375	27376	27377	27383
24	27384	27385	27387	27389	27390	27391	27392	27393	27399
25	27400	27401	27403	27405	27406	27407	27408	27409	27415
26	27416	27417	27419	27421	27422	27423	27424	27425	27431
27	27432	27433	27435	27437	27438	27439	27440	27441	27447
28	27448	27449	27451	27453	27454	27455	27456	27457	27463
29	27464	27465	27467	27469	27470	27471	27472	27473	27479
30	27480	27481	27483	27485	27486	27487	27488	27489	27495
31	27496	27497	27499	27501	27502	27503	27504	27505	27511

Πίνακας 3 Διευθύνσεις χαρακτηριστικών των 32 sprites

Ο χώρος που καταλαμβάνει κάθε sprite στον πίνακα είναι 16 bytes. Όπως μπορείτε να δείτε, οι συντεταγμένες X και Y είναι αριθμοί των 2byte. Τα sprites δέχονται αρνητικές συντεταγμένες, οπότε μπορείτε να εκτυπώσετε μερικώς ένα sprite στην οθόνη, δίνοντας την εντύπωση ότι έρχεται bit προς bit. Δεν μπορείτε να ορίσετε αρνητικές συντεταγμένες με την εντολή POKE, αλλά μπορείτε με την εντολή |LOCATESP και επίσης με την εντολή |POKE, η οποία είναι μια έκδοση της εντολής POKE της BASIC, αλλά δέχεται αρνητικούς αριθμούς (και αριθμούς των 16 bit).

Είναι καλή πρακτική να τοποθετείτε τον χαρακτήρα ή το διαστημόπλοιο στη θέση 31 (υπάρχουν 32 sprites αριθμημένα από το 0 έως το 31). Εάν το διαστημόπλοιο σας έχει τη θέση 31, θα εκτυπωθεί τελευταίο, πάνω από τα υπόλοιπα sprites σε περίπτωση επικάλυψης.

8.7 Όλα τα sprites εκτυπώνονται και ταξινομούνται

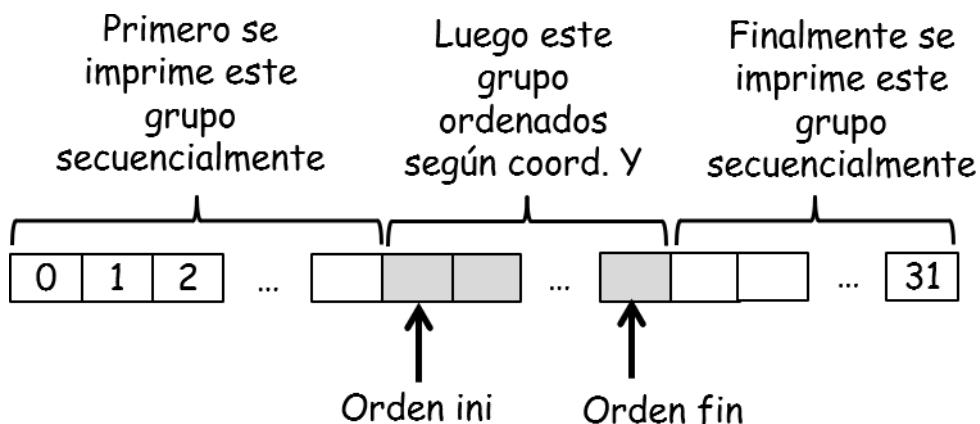
Στη βιβλιοθήκη 8BP υπάρχει μια εντολή που εκτυπώνει όλα τα sprites που έχουν ενεργή τη σημαία εκτύπωσης ταυτόχρονα. Αυτή είναι η εντολή |PRINTSPALL

Αυτή η εντολή έχει 4 παραμέτρους, αλλά χρειάζεται να τις συμπληρώσετε μόνο την πρώτη φορά που θα την καλέσετε, επειδή τις επόμενες φορές θα θυμάται τις παραμέτρους και θα χρειάζεται να τις συμπληρώσετε ξανά μόνο αν θέλετε να αλλάξετε κάποια από αυτές. Αυτό είναι χρήσιμο, επειδή το πέρασμα παραμέτρων είναι πολύ χρονοβόρο (μπορείτε να εξοικονομήσετε περισσότερο από 1ms αποφεύγοντας το πέρασμα παραμέτρων).

Οι παράμετροι είναι:

|PRINTSPALL, <ordenini>, <ordenfin>, <animate>, <synchronism>.

- **Η παράμετρος sync** μπορεί να λάβει τις τιμές 0 ή 1 και υποδεικνύει ότι θα περιμένει τη διακοπή της σάρωσης οθόνης πριν από την εκτύπωση. Αν θέλετε ταχύτητα, δεν τη συνιστώ. Αν θέλετε μεγαλύτερη ομαλότητα, ίσως ναι.
- **Η παράμετρος animation** μπορεί να πάρει τις τιμές 0 ή 1. Αν είναι ενεργή, πριν από την εκτύπωση κάθε sprite, η σημαία animation θα ελέγχεται στο status byte και αν είναι ενεργή, τότε θα μεταβαίνει στο επόμενο καρέ. Αυτό είναι πολύ χρήσιμο με τους εχθρούς, αλλά με τον χαρακτήρα σας ίσως όχι, καθώς μπορεί να θέλετε να τον κινείτε μόνο όταν τον μετακινείτε και όχι σε κάθε καρέ. ΣΗΜΑΝΤΙΚΟ: Το animation γίνεται πριν την εκτύπωση, όχι μετά την εκτύπωση. Αυτό σημαίνει ότι αν έχετε μόλις εκχωρήσει μια ακολουθία κινούμενων σχεδίων, δεν θα δείτε το πρώτο καρέ της ακολουθίας αυτής.
- **Οι παράμετροι τάξης ("ordenini", "ordenfin")** υποδεικνύουν το αρχικό και το τελικό sprite που καθορίζουν την ομάδα sprites ταξινομημένων με βάση τη συντεταγμένη "Y" που πρόκειται να εκτυπώσουμε. Για παράδειγμα, αν δώσουμε τις τιμές 0,0 τότε θα εκτυπωθούν διαδοχικά από το sprite 0 έως το sprite 31. Αν δώσουμε τις τιμές 0,8 θα εκτυπωθούν από το 0 έως το 8 διατεταγμένα (9 sprites) και από το 10 έως το 31 διαδοχικά. Εάν ορίσουμε 0,31 όλα τα sprites θα εκτυπωθούν με τη σειρά. Αν ορίσουμε ένα 10,20 τα sprites θα εκτυπωθούν διαδοχικά από το 0 έως το 9, στη συνέχεια θα εκτυπωθούν διατεταγμένα από το 10 έως το 20 και τέλος θα εκτυπωθούν διαδοχικά από το 21 έως το 31.



Σχ. 47 Διαδοχικές και διατεταγμένες ομάδες sprite

Η ταξινόμηση είναι πολύ χρήσιμη για την κατασκευή παιχνιδιών όπως το "Renegade" ή το "Golden AXE", όπου είναι απαραίτητο να δοθεί ένα εφέ βάθους. Η διάταξη εκτιμάται όταν υπάρχουν επικαλύψεις μεταξύ των sprites.



Σχ. 48 Επίδραση της διάταξης των sprites

- | PRINTSPALL, 0,0,1,0 : εκτυπώνει διαδοχικά όλα τα sprites
- | PRINTSPALL, 0,31,1,0 : εκτυπώνει όλα τα sprites με σειρά
- | PRINTSPALL, 0,7,1,0: εκτύπωση 8 διατεταγμένων και των υπολοίπων διαδοχικά
- | PRINTSPALL, 16,24,1,0: 16 διαδοχικά, 9 ταξινομημένα και 7 διαδοχικά

Εάν η παράμετρος "ordenini" παραλείπεται, λαμβάνεται υπόψη η τελευταία τιμή που έχει εκχωρηθεί, ή μηδέν εάν δεν έχει ποτέ εκχωρηθεί καμία τιμή. Επίσης, αν πρόκειται να τροποποιήσετε κάποια από τις δύο παραμέτρους ταξινόμησης, είναι καλή ιδέα να εκτελέσετε πρώτα το PRINTSPALL,0,0,0,0,0,0,0,0,0, ώστε τα sprites να αναδιαταχθούν πρώτα διαδοχικά πριν τα ταξινομήσετε με μια νέα διαμόρφωση.

Η εκτύπωση με τη σειρά είναι πιο δαπανηρή υπολογιστικά από την εκτύπωση με τη σειρά. Αν έχετε μόνο 5 sprites που πρέπει να ταξινομηθούν, περάστε για παράδειγμα ένα 0,4 ως παράμετροι ταξινόμησης, μην περάστε 0,31. Η ταξινόμηση όλων των sprites διαρκεί περίπου 2,5 ms, αλλά αν ταξινομήσετε μόνο 5 sprites μπορείτε να εξουκονομήσετε 2 ms. Ίσως έχετε πολλά sprites και δεν αξίζει να ταξινομήσετε κάποια από αυτά, όπως τα πλάνα ή τα sprites που ξέρετε ότι δεν θα επικαλύπτονται.

Ο αλγόριθμος που χρησιμοποιείται για την ταξινόμηση των sprites είναι μια παραλλαγή του λεγόμενου αλγορίθμου "φούσκα". Αν και θα βρείτε στη βιβλιογραφία ότι ο λεγόμενος αλγόριθμος "φούσκα" είναι πολύ αναποτελεσματικός, αυτό το λένε όσοι μιλούν για μια λίστα τυχαίων αριθμών προς ταξινόμηση. Πρόκειται για μια περίπτωση όπου τα sprites είναι κανονικά σχεδόν ταξινομημένα και από το ένα καρέ στο επόμενο μόνο ένα ή δύο sprites είναι αταξινόμητα, όχι περισσότερα, καθώς οι συντεταγμένες τους εξελίσσονται "ομαλά". Έτσι, ο αλγόριθμος περνάει από τη λίστα των sprites και όταν βρει μερικά ατακτοποίητα sprites, τα γυρίζει και σταματάει την ταξινόμηση. Είναι εξαιρετικά γρήγορος, και ακόμη και αν είναι σε θέση να ταξινομήσει μόνο ένα ζευγάρι sprites κάθε φορά, είναι ιδανικός για αυτή την περίπτωση χρήσης. Μόνο στην περίπτωση που υπάρχουν 2 μη ταξινομημένα sprites και επικαλύπτονται, θα υπάρξει ένα καρέ όπου θα δούμε ένα από αυτά να εκτυπώνεται εκτός σειράς, αλλά θα διορθωθεί στο επόμενο καρέ. Είναι ανεπαίσθητο.

Μερικές φορές μπορεί να θέλετε η ταξινόμηση να είναι πλήρης σε κάθε καρέ. Δηλαδή, όχι για να έχετε μερικά sprites ταξινομημένα σε κάθε κλήση του |PRINTSPALL, αλλά για να είστε σίγουροι ότι είναι όλα ταξινομημένα. Η βιβλιοθήκη **8BP** σας επιτρέπει να

το κάνετε αυτό μέσω των τεσσάρων τρόπων ταξινόμησης, τους οποίους μπορείτε να ορίσετε καλώντας την εντολή

|**Το παρακάτω είναι ένα PRINTSPALL με μία μόνο παράμετρο (εκτελέστε το μόνο μία φορά για να ορίσετε τον τρόπο ταξινόμησης):**

PRINTSPALL,0 : μερική ταξινόμηση με χρήση Ymin

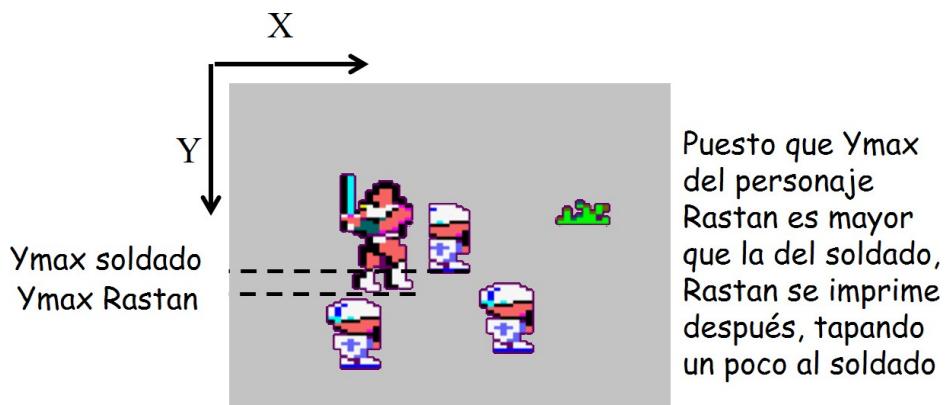
PRINTSPALL,1 : πλήρης ταξινόμηση με χρήση Ymin

PRINTSPALL,2 : μερική ταξινόμηση με χρήση Ymax

PRINTSPALL,3 : πλήρης ταξινόμηση με χρήση Ymax

Η ταξινόμηση με χρήση της Ymax βασίζεται στη μεγαλύτερη συντεταγμένη Y των sprites, δηλαδή στο σημείο όπου βρίσκονται τα πόδια τους και όχι τα κεφάλια τους. Αν τα sprites έχουν το ίδιο μέγεθος, η ταξινόμηση με βάση την Ymin μπορεί να λειτουργήσει, αλλά αν τα sprites έχουν διαφορετικό ύψος, μπορεί να θέλετε να ταξινομήσετε σύμφωνα με το πού βρίσκονται τα πόδια κάθε χαρακτήρα, και γι' αυτό θα πρέπει να χρησιμοποιήσετε τη λειτουργία ταξινόμησης 2 ή 3.

Οι λειτουργίες ταξινόμησης Ymax είναι πιο αργές, περίπου 0,128 ms ανά sprite, οπότε χρησιμοποιήστε τις όταν τις χρειάζεστε πραγματικά.



Σχ. 49 Ταξινόμηση σύμφωνα με το Ymax

Η πλήρης ταξινόμηση καταναλώνει πολύ λίγο περισσότερο από τη μερική ταξινόμηση (περίπου 0,3ms). Αυτό οφείλεται στο γεγονός ότι τα sprites δεν είναι σχεδόν ποτέ ανακατεμένα από το ένα καρέ στο άλλο, αλλά ακόμη και αυτά τα 0,3ms αξίζει να τα εξοικονομήσετε αν είναι δυνατόν.

Θυμηθείτε ότι η εντολή PRINTSPALL έχει "μνήμη", οπότε αρκεί να την καλέσουμε την πρώτη φορά με παραμέτρους και από εκείνη τη στιγμή και μετά μπορούμε να καλέσουμε την PRINTSPALL χωρίς παραμέτρους, επειδή η εντολή "κρατάει" τις τιμές των παραμέτρων με τις οποίες κλήθηκε και δεν χρειάζεται να της τις δώσουμε, εκτός αν αλλάξουν. Αυτό εξοικονομεί περισσότερα από 1ms, αφού ο αναλυτής εργάζεται λιγότερο.

8.8 Συγκρούσεις μεταξύ sprites

Για να ελέγξετε αν ο χαρακτήρας σας ή η βολή σας έχει συγκρουστεί με άλλα sprites μπορείτε να χρησιμοποιήσετε την εντολή

|COLSP, <αριθμός_στίχων>, @collision%.

Όπου ο αριθμός sprite είναι το sprite που θέλετε να ελέγξετε (ο χαρακτήρας σας ή το πλάνο σας) και η μεταβλητή "collision" είναι μια ακέραια μεταβλητή που έπρεπε

προηγουμένως να οριστεί, αναθέτοντας μια αρχική τιμή, για παράδειγμα:

σύγκρουση% = 0

|COLSP, 1, @collision%, 1, @collision%, 1, @collision%.

Η μεταβλητή "collision" θα γεμίσει με το πρώτο αναγνωριστικό sprite που εντοπίζεται να έχει συγκρουστεί με το sprite σας, αν και μπορεί να συμβούν πολλαπλές συγκρούσεις, αλλά η εντολή σας δίνει μόνο ένα αποτέλεσμα.

Εσωτερικά, η βιβλιοθήκη 8BP διατρέχει τα sprites που συγκρούονται από το 31 έως το 0 (τα τρέχει με αντίστροφη σειρά), και αν έχουν ενεργή τη σημαία σύγκρουσης (bit 1 του status byte) τότε ελέγχει αν συγκρούεται με το sprite σας. Αν δεν υπάρχει σύγκρουση με κανένα από αυτά, η μεταβλητή collision% παίρνει την τιμή 32. Αν υπάρχει, επιστρέφει τον αριθμό του sprite που συγκρούεται με το δικό σας sprite. Αν για παράδειγμα συγκρουστούν το 4 και το 12, η συνάρτηση θα επιστρέψει το 12 επειδή ελέγχει το 12 πριν από το 4.

Ούτε ο χαρακτήρας σας ούτε η βολή σας πρέπει να έχουν ταυτόχρονα ενεργές τις σημαίες "collider" (bit 1) και "collider" (bit 5), αλλιώς θα συγκρούονται πάντα... με τον εαυτό τους! Δηλαδή, ένα sprite δεν μπορεί να έχει ενεργά τα bit 1 και 5 ταυτόχρονα.

Η σύγκρουση μεταξύ sprites είναι μια δαπανηρή εργασία. Εσωτερικά η βιβλιοθήκη πρέπει να υπολογίσει την τομή μεταξύ των ορθογωνίων που περιέχουν κάθε sprite για να καθορίσει αν υπάρχει επικάλυψη μεταξύ τους. Για να εξοικονομήσετε υπολογισμούς, είναι καλύτερο να τοποθετείτε τους εχθρούς σε διαδοχικές θέσεις sprite. Αν για παράδειγμα οι εχθροί με τους οποίους μπορούμε να συγκρουστούμε είναι τα sprites 15 έως 25, μπορούμε να ορίσουμε τη σύγκρουση να ελέγχει μόνο αυτά τα sprites. Για να το κάνουμε αυτό, καλούμε τη σύγκρουση στο sprite 32, το οποίο δεν υπάρχει. Αυτό θα πει στη βιβλιοθήκη 8BP ότι πρόκειται για πληροφορίες διαμόρφωσης για την εντολή, υποδεικνύοντας το εύρος των collisionible sprites που πρέπει να ελεγχθούν για κάθε collider:

|COLSP, 32, <πρωτογενής αρχικός>, <πρωτογενής τελικός>.

Παράδειγμα:

|COLSP, 32, 15, 25

Αυτή η βελτιστοποίηση δεν είναι πολύ σημαντική, αλλά γίνεται σημαντική όταν το COLSP καλείται πολλές φορές ή όταν χρησιμοποιείται η εντολή |COLSPALL, η οποία εσωτερικά καλεί το COLSP πολλές φορές.

Μια άλλη ενδιαφέρουσα βελτιστοποίηση, ικανή να εξοικονομήσει 1 χιλιοστό του δευτερολέπτου σε κάθε κλήση, είναι να πείτε στην εντολή να χρησιμοποιεί πάντα την ίδια μεταβλητή BASIC για να αφήσει το αποτέλεσμα της σύγκρουσης. Για να το κάνουμε αυτό, θα την υποδείξουμε χρησιμοποιώντας την 33 ως sprite, η οποία επίσης δεν υπάρχει.

col%=0

|COLSP, 33, @col%, @col%, @COLSP, 33, @col%, @COLSP, 33, @col%

Μόλις εκτελεστούν αυτές οι δύο γραμμές, οι επόμενες κλήσεις της COLSP θα αφήσουν το αποτέλεσμα στη μεταβλητή col, χωρίς να χρειάζεται να το υποδείξετε, για παράδειγμα:

|COLSP, 23 : REM αυτή η κλήση είναι ισοδύναμη με |COLSP, 23, @col%.

ΣΗΜΑΝΤΙΚΟ: Η μεταβλητή σύγκρουσης στην εντολή **|COLSP** δεν είναι αυτή που χρησιμοποιείται στην εντολή **|COLSPALL**. Πρόκειται για διαφορετικές μεταβλητές (εκτός αν δώσετε και στις δύο εντολές την ίδια μεταβλητή για να ενεργήσετε σε αυτήν).

8.9 Ρύθμιση της ευαισθησίας σύγκρουσης sprite

Είναι δυνατόν να ρυθμίσετε την ευαισθησία της εντολής COLSP, αποφασίζοντας αν η επικάλυψη μεταξύ των sprites πρέπει να είναι αρκετά pixel ή μόνο ένα pixel, προκειμένου να θεωρηθεί ότι έχει συμβεί σύγκρουση.

Αυτό μπορεί να γίνει με τον καθορισμό των αριθμού των εικονοστοιχείων (εικονοστοιχεία στην κατεύθυνση Y, bytes στην κατεύθυνση X) της επικάλυψης που απαιτείται τόσο στην κατεύθυνση Y όσο και στην κατεύθυνση X, χρησιμοποιώντας την εντολή COLSP και καθορίζοντας το sprite 34 (το οποίο δεν υπάρχει).

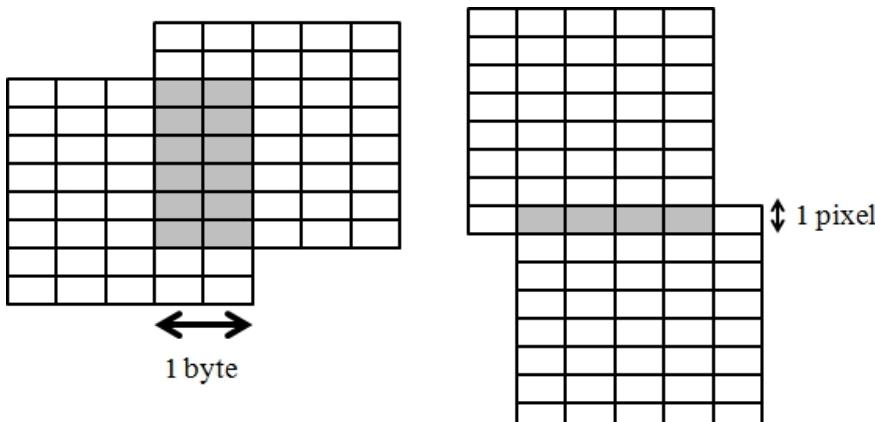
| **COLSP, 34, <dy>, <dx>, <dx>, <dx>, <dx>, <dx>, <dx>, <dx>**

Η βιβλιοθήκη 8BP δεν χρησιμοποιεί "εικονοστοιχεία" στη συντεταγμένη X, αλλά bytes, οπότε πρέπει να λάβετε υπόψη σας ότι μια σύγκρουση 1 byte είναι στην πραγματικότητα 2 εικονοστοιχεία, και αυτή είναι η ελάχιστη δυνατή σύγκρουση όταν ορίζετε dx=0.

Στη συντεταγμένη Y, η βιβλιοθήκη λειτουργεί με γραμμές, έτσι ώστε dy=0 να σημαίνει σύγκρουση ενός μόνο εικονοστοιχείου.

Μια αυστηρή σύγκρουση, χρήσιμη για πυροβολισμούς, θα ήταν αυτή που δεν ανέχεται κανένα περιθώριο, θεωρώντας σύγκρουση μόλις υπάρξει μια ελάχιστη επικάλυψη μεταξύ των sprites (1 pixel στην κατεύθυνση Y ή ένα byte στην κατεύθυνση X).

| **COLSP, 34, 0, 0, 0: rem σύγκρουση μόλις υπάρξει ελάχιστη επικάλυψη**



Σχ. 50 Αντηρή σύγκρουση με COLSP, 34, 0, 0, 0

Ωστόσο, αν φτιάχνουμε ένα παιχνίδι σε MODE 0, όπου τα pixel είναι πιο πλατιά από ό,τι ψηλά, είναι ίσως πιο σωστό να δώσουμε κάποιο περιθώριο στον άξονα Y, αλλά κανένα στον άξονα X.

X. Για παράδειγμα:

| **COLSP, 34, 2, 0 : rem σύγκρουση με 3 rīx στο Y και 1 byte στο X**

Η σύστασή μου είναι ότι, αν υπάρχουν στενά ή μικρά πλάνα, ρυθμίστε την σύγκρουση σε (dy=1, dx=0), ενώ αν υπάρχουν μόνο μεγάλοι χαρακτήρες μπορείτε να το αφήσετε με μεγαλύτερο περιθώριο (dy=2, dx=1). Θα πρέπει επίσης να λάβετε υπόψη σας ότι, αν τα sprites σας έχουν ένα "περιθώριο" διαγραφής γύρω τους για να κινούνται και να διαγράφονται, αυτό το περιθώριο δεν θα πρέπει να αποτελεί μέρος της εξέτασης της

σύγκρουσης, οπότε είναι λογικό τόσο το dy όσο και το dx να μην είναι μη μηδενικά. Σε κάθε περίπτωση, αυτό είναι κάτι που θα αποφασίσετε ανάλογα με το είδος του παιχνιδιού που φτιάχνετε.

8.10 Ποιος συγκρούεται και με ποιον: COLSPALL

Με τη λειτουργία **|COLSP** που είδαμε μέχρι τώρα, είναι δυνατή η ανίχνευση σύγκρουσης ενός sprite με όλα τα άλλα. Ωστόσο, αν έχουμε πολλαπλές βολές, όπου για παράδειγμα το σκάφος μας μπορεί να ρίξει μέχρι και 3 βολές ταυτόχρονα, θα πρέπει να ανιχνεύσουμε τη σύγκρουση του καθενός από αυτά και επιπλέον του σκάφους μας, με αποτέλεσμα 4 κλήσεις στην **|COLSP**.

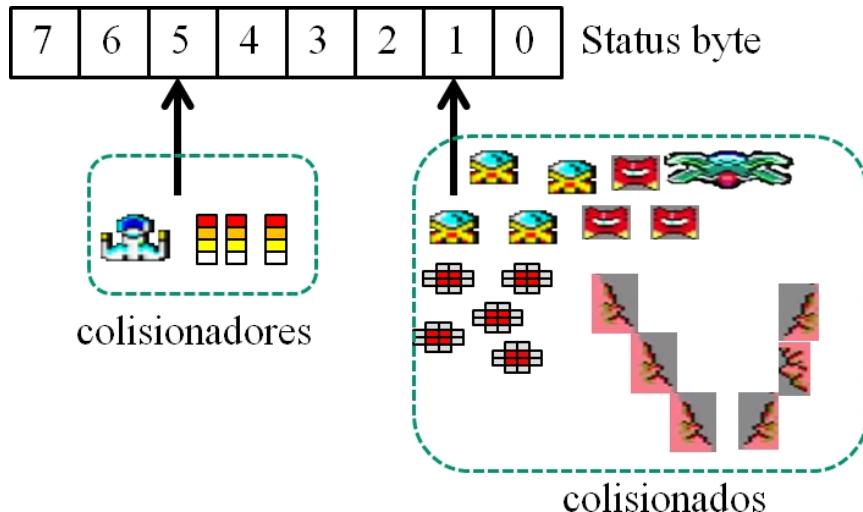
Λάβετε υπόψη ότι κάθε κλήση περνάει από το επίπεδο ανάλυσης, οπότε τέσσερις κλήσεις είναι δαπανηρές. Για αυτό έχουμε μια πολύ ισχυρή εντολή: **|COLSPALL**.

Αυτή η συνάρτηση λειτουργεί σε δύο βήματα: πρώτα πρέπει να καθορίσουμε ποιες μεταβλητές θα αποθηκεύουν τον συγκρουστή και το sprite που συγκρούστηκε. Η ακόλουθη εντολή θα εκτελεστεί μόνο μία φορά και χρησιμεύει για να ορίσουμε τις μεταβλητές στις οποίες θα λάβουμε τα αποτελέσματα, οι οποίες πρέπει να υπάρχουν προηγουμένως:

|COLSPALL, @collider%, @collider%, @collider%.

Και στη συνέχεια, σε κάθε κύκλο παιχνιδιού, απλά καλούμε τη συνάρτηση **|COLSPALL** χωρίς παραμέτρους.

Η συνάρτηση θα θεωρεί ως "συγκρουόμενα" sprites εκείνα που έχουν τη σημαία σύγκρουσης σε "1" στο byte κατάστασης (bit 5), και ως "συγκρουόμενα" εκείνα τα sprites που έχουν τη σημαία σύγκρουσης (bit 1) στο byte κατάστασης σε "1". Τα συγκρουόμενα sprites θα πρέπει να είναι το πλοίο μας και οι βολές μας, και τα συγκρουόμενα sprites θα πρέπει να είναι όλα εκείνα με τα οποία μπορούμε να συγκρουστούμε: εχθρικά πλοία και βολές, βουνά, κλπ. Όπως είπα και πριν, ένα sprite δεν πρέπει να έχει και τα δύο bits ενεργά (=1) ταυτόχρονα.



Σχ. 51 Συγκρουστήρες έναντι συγκρουστήρων

Η συνάρτηση **|COLSPALL** ξεκινάει ελέγχοντας το sprite 31 (αν είναι collider) και προχωράει προς τα κάτω μέχρι το sprite 0, καλώντας εσωτερικά το **|COLSP** για κάθε collider sprite. Για κάθε collider, τα collider sprites επίσης διατρέχονται με φθίνουσα σειρά (από το 31 στο 0). Μόλις ανιχνεύσει σύγκρουση, διακόπτει την εκτέλεσή της και επιστρέφει την τιμή του collider και του collider. Επομένως, είναι σημαντικό το πλοίο μας να έχει υψηλότερο sprite από τις βολές μας. Με αυτόν τον τρόπο, αν χτυπηθούμε

Θα το ανιχνεύσουμε, ακόμη και αν έχουμε χτυπήσει έναν εχθρό με μια βολή την ίδια στιγμή.

Σε κάθε κύκλο παιχνιδιού μπορεί να ανιχνευθεί μόνο μία σύγκρουση, αλλά αυτό είναι αρκετό. Δεν αποτελεί σημαντικό περιορισμό το γεγονός ότι σε κάθε καρέ μόνο ένας εχθρός μπορεί να αρχίσει να "εκρήγγνυται". Εάν, για παράδειγμα, ρίξετε μια χειροβομβίδα και υπάρχει μια ομάδα 5 στρατιωτών που επηρεάζονται, κάθε στρατιώτης θα αρχίσει να πεθαίνει σε διαφορετικό καρέ και μετά από 5 καρέ θα εκραγούν όλοι. Χρησιμοποιώντας το |COLSPALL δεν θα ανατιναχτούν όλοι ταυτόχρονα, αλλά το παιχνίδι σας θα είναι ταχύτερο και σε ένα παιχνίδι arcade αυτό είναι πολύ σημαντικό.

Σε περίπτωση κλήσης του |COLSPALL με μία μόνο παράμετρο,

|COLSPALL, <αρχικός επιταχυντής>

Οι συγκρουστές θα σαρώνονται από τον υποδεικνυόμενο συγκρουστή -1 έως το sprite μηδέν, με φθίνουσα σειρά. Έτσι, αν πρέπει να ανιχνεύσετε περισσότερες από μία συγκρούσεις ανά κύκλο παιχνιδιού, μπορείτε να το κάνετε με διαδοχικές επικλήσεις COLSPALL, <collider> μέχρι η μεταβλητή collider να πάρει την τιμή 32.

Παράδειγμα:

|COLSPALL, 7 : αναζητήσεις rem για συγκρούσεις από τον επιταχυντή
6

8.10.1 Πώς να προγραμματίστε μια πολλαπλή λήψη με το

COLSPALL Το πρώτο πράγμα που πρέπει να κάνετε είναι να αποφασίσετε πόσες ενεργές λήψεις μπορούν να υπάρξουν. Αν αποφασίσετε ότι το σκάφος σας μπορεί να εκτοξεύσει 3 βλήματα ταυτόχρονα, τότε θα πρέπει να αποθεματικό 3 αναγνωριστικά sprite για ενεργοποίηση. Στη συνέχεια, πρέπει να ρυθμίσετε την καθυστέρηση μεταξύ της μιας βολής και της επόμενης για να αποτρέψετε δύο βλήματα να χτυπήσουν σχεδόν το ένα το άλλο, αν πυροβολήσετε πολύ γρήγορα. Αυτό μπορεί να γίνει ορίζοντας μια ελάχιστη καθυστέρηση μεταξύ των βολών.

Στο ακόλουθο παράδειγμα έχω ορίσει μια καθυστέρηση μεταξύ των βολών 10 κύκλων παιχνιδιού. Για να γίνει αυτό, πατώντας το πλήκτρο διαστήματος (πλήκτρο 47) ελέγχεται αν έχουν περάσει τουλάχιστον 10 κύκλοι από την τελευταία βολή. Εάν όχι, δεν θα πυροβολήσει.

130 ----- "κύκλος παιχνιδιού".

150 |AUTOALL,1:|PRINTSPALL,0,1,0

170 ' character movement routine -----

AN INKEY(47)=0 TOTE AN delay<cycle-10 TOTE delay=cycle:disp= 1+
disp MOD 3 :|LOCATESP,10+disp,PEEK(27001)+8,PEEK(27003):

|SETUPSP,10+disp,0,137: |SETUPSP,10+disp,15,3+dir

ΕΑΝ INKEY(27)=0 dir=0:|SETUPSP,0,6,1:'go right
TOTE

190 ΕΑΝ INKEY(34)=0 dir=1:|SETUPSP,0,6,-1:'go left
TOTE

193 κύκλος=κύκλος+
1

310 GOTO 150

Για να επιλέξετε το sprite που θα χρησιμοποιηθεί ως σκανδάλη, εκτελείται η εντολή:

disp = 1+ disp Mod 3

Αυτό θα επιτρέψει στη βολή σας να πάρει τις τιμές 1,2,3,4 εναλλακτικά. Αν θέλετε τα sprites να είναι 20,21,22,23 μπορείτε να χρησιμοποιήσετε **disp = 20 + disp Mod 3**.

γιατί αν βάλετε το 21 ως άθροισμα, δεν λειτουργεί (δοκιμάστε το μόνοι σας). Αυτό είναι το θέμα με τη σπονδυλωτή αριθμητική.

Και τώρα ερχόμαστε στις συγκρούσεις και στο πλεονέκτημα της χρήσης του COLSPALL, που είναι πολύ πιο γρήγορο από την επίκληση του COLSP πολλές φορές. Οι μόνες σημαντικές συστάσεις είναι οι εξής:

- Ας αφήσουμε τον αριθμό <sprite> να είναι μεγαλύτερος από τους ενεργοποιητές μας, έτσι ώστε |COLSPALL ελέγξτε το πριν από τη βολή.
- Ότι έχουμε ρυθμίσει το |COLSP να ελέγχει μόνο τη λίστα των sprites που είναι εχθροί και πρέπει να συγκρουστούν, χρησιμοποιώντας το |COLSP 32, <start>, <end>.

Πριν από την έναρξη του κύκλου παιχνιδιού ορίζουμε τις μεταβλητές μας:

collider=32:collided=32:|COLSPALL,@collider,@collided

Στον κύκλο του παιχνιδιού θα βάλουμε:

**|COLSPALL:IF collider<32 THEN if collider=31 THEN GOSUB 300:goto 2000:
ELSE GOSUB 770**

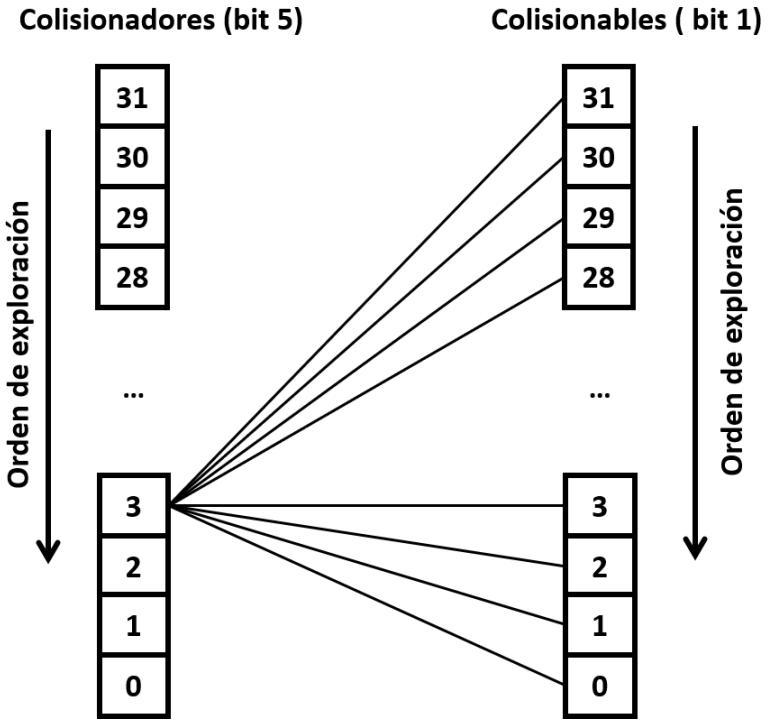
Με αυτή τη γραμμή ξέρουμε ήδη αν υπάρχει σύγκρουση, γιατί τότε η μεταβλητή "collider" θα είναι <32 Επιπλέον, αν είναι ίση με 31 τότε είναι το σκάφος μας (έχουμε χτυπηθεί) και αν όχι, τότε σίγουρα μια από τις βολές μας έχει χτυπήσει ένα εχθρικό σκάφος και θα πάμε στη ρουτίνα που βρίσκεται στη γραμμή 770, όπου θα βρούμε κάτι τέτοιο:

**769' --- Πυροβολισμός σύγκρουσης ρουτίνας -----
770 |SETUPSP, collider, 9, img deleted:'associate deleted image to the
shot
772 |PRINTSP, collider: 'Διαγράφουμε το πλάνο
775 |SETUPSP, collider, 0, 0: 'Απενεργοποίηση ενεργοποίησης
777 if collided>=duros then return:'εχθρός άφθαρτος
778 ' Η ακολουθία 4 είναι μια ακολουθία κινουμένων σχεδίων του
"Θανάτου", ενός
έκρηξη
780 |SETUPSP, collided, 7, 4:|SETUPSP, collided, 0, &x101: return**

Εν ολίγοις, με μία μόνο κλήση της COLSPALL γνωρίζουμε ποιος έχει συγκρουστεί ("collider") και με ποιον έχει συγκρουστεί ("collided").

8.10.2 Ποιος συγκρούεται όταν υπάρχουν πολλές επικαλύψεις

Είναι πολύ σημαντικό να έχετε κατά νου ότι η 8BP διατρέχει τα collider από το 31 έως το 0 και για κάθε ένα από αυτά, διατρέχει τα colliderables από το 31 έως το 0. Πρέπει να συσχετίσουμε τα Sprite IDs με τα sprites μας ανάλογα με το πώς θέλουμε να συγκρούονται.



Σχ. 52 Σειρά ελέγχου σύγκρουσης

Αν το sprite μας (collider) συγκρουστεί με δύο sprites στην ίδια περιοχή, μπορούμε να γνωρίζουμε εκ των προτέρων με ποιο από αυτά θα συγκρουστούμε, κάτι που είναι πολύ χρήσιμο για ορισμένα παιχνίδια.

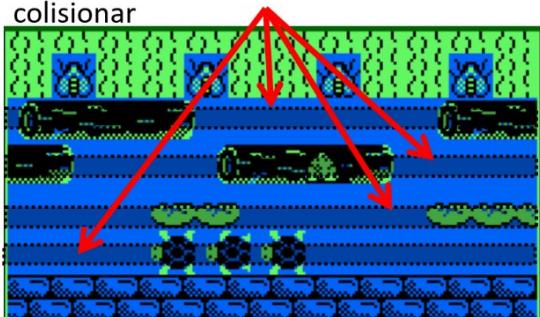
Ας υποθέσουμε το παιχνίδι "frogger", στο οποίο ένας βάτραχος πρέπει να διασχίσει ένα ποτάμι πηδώντας πάνω από κορμούς. Αν η σύγκρουση είναι πάνω σε ένα κούτσουρο, δεν θα πεθάνουμε, αλλά αν η σύγκρουση είναι πάνω σε ένα ποτάμι, θα πεθάνουμε.

Για να το προγραμματίσουμε μπορούμε να βάλουμε 4 ποτάμια (4 ακίνητα επιμήκη sprites) και πάνω σε αυτά να μετακινήσουμε κάποια sprites που είναι κορμοί. Θα μπορούσαμε να δημιουργήσουμε την ακόλουθη κατανομή:

- Ο βάτραχος είναι το sprite 31 (υποθέστε ότι είναι συγκρουόμενος).
- Οι κορμοί είναι τα sprites 4, 5, 6, 7 (συγκρούσεις).
- Τα ποτάμια είναι sprites 0, 1, 2, 3 (collidable).

Οι ποταμοί μπορούν να έχουν απενεργοποιημένη τη σημαία εκτύπωσης, ώστε να μπορούν να συγκρουστούν με το βάτραχο (σημαία σύγκρουσης) χωρίς να εκτυπωθούν. Δηλαδή, θα τους θέσουμε την κατάσταση =2

Ríos (rectángulos largos) que no se imprimen pero que están ahí y pueden colisionar



Se detecta la colisión con el tronco antes que con el río porque el río (collided) tiene un Sprite ID menor que el tronco



Εικ. 53 σε περίπτωση επικάλυψης μας ενδιαφέρει η σύγκρουση του κορμού.

Λοιπόν, καθώς οι κορμοί και το ποτάμι επικαλύπτονται, τη στιγμή που ο βάτραχος σκαρφαλώνει σε έναν κορμό συγκρούεται και με τους δύο, αλλά ο κορμός ελέγχεται πρώτα, καθώς έχει υψηλότερο αναγνωριστικό sprite. Η εντολή σύγκρουσης θα ανιχνεύσει μόνο τη σύγκρουση με τον κορμό. Αντίθετα, αν ο βάτραχος πηδήξει πάνω από το νερό, τότε η εντολή σύγκρουσης θα ανιχνεύσει το ποτάμι και αφού αξιολογήσει από τη BASIC τη μεταβλητή "collided" και δει ότι πρόκειται για ποτάμι, θα διαπιστώσουμε ότι ο βάτραχος μας πρέπει να πεθάνει.

8.10.3 Προηγμένη χρήση του byte κατάστασης σε συγκρούσεις

Μερικές φορές μπορεί να θέλετε ένας εχθρός να μην σας σκοτώσει συγκρουόμενος με τον χαρακτήρα σας επειδή βρίσκεται σε ειδική κατάσταση ή επειδή είναι απλά μακριά σε ένα παιχνίδι που προσποιείται ότι οι εχθροί πλησιάζουν.

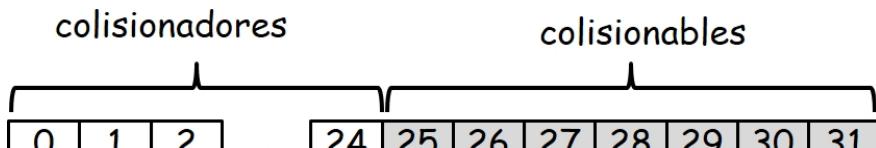
Ορισμένες ειδικές περιστάσεις μπορεί να απαιτούν τη χρήση ενός "σήματος" στο sprite για να υποδείξει ότι, παρόλο που υπήρξε σύγκρουση, δεν πρέπει να πεθάνει ή να σας σκοτώσει.

Για το σκοπό αυτό μπορείτε να χρησιμοποιήσετε τις σημαίες που δεν χρησιμοποιείτε στο sprite και να τις ελέγχετε στη ρουτίνα σύγκρουσης.

Ας δούμε ένα παράδειγμα για έναν εχθρό που θέλουμε να κάνουμε ακίνδυνο επειδή είναι μακριά (προσομοιάζοντας 3D) ή με χαμηλή ενέργεια αλλά συγκρούεται μαζί μας.

Ας υποθέσουμε ότι ο χαρακτήρας σας συγκρούεται και ο εχθρός είναι ένας συγκρουόμενος. Μπορείτε να επιβάλλετε ότι η σύγκρουση ανιχνεύεται μόνο με sprites μεγαλύτερα από 25 και αν ο εχθρός είναι νούμερο 20 (για παράδειγμα) δεν θα μπορεί να συγκρουστεί με τον εαυτό του.

|COLSP,32,25,31



Σχήμα 54 επίδραση της COLSP,32,25,31

Ως "ακίνδυνη" ένδειξη θα χρησιμοποιήσουμε τη σημαία collided, θέτοντάς την σε 1. Ετσι, θα θέσουμε τη σημαία enemy (sprite 20) σε 1 στο byte κατάστασης.

Ας υποθέσουμε τώρα ότι η εντολή |COLSPALL ανιχνεύει μια σύγκρουση και αφήνει το αποτέλεσμα στο collider και collided.

|COLSPALL

Εάν collider<32 τότε GOSUB 100

<Οδηγίες>

100 rem ρουτίνα σύγκρουσης

110 dir=27000 + collider*16 :rem byte διεύθυνσης κατάστασης του collider

120 εάν PEEK (dir) και 2 THEN RETURN: rem ακίνδυνο εάν το bit
collided=1

130 <Ένας εχθρός συγκρούστηκε και δεν είναι ακίνδυνος>.

Η εντολή "if PEEK (dir) and 2" είναι αυτή που ελέγχει το συγκρουόμενο bit, αφού το 2 στο δυαδικό σύστημα είναι 00000010, απλώς η θέση του bit αυτού στην

κατάσταση sprite.

Όταν ο εχθρός δεν είναι πλέον ακίνδυνος, απλά θέτουμε τη σημαία σύγκρουσης στο 0 και σε περίπτωση περαιτέρω σύγκρουσης, θα μας σκοτώσει.

Αυτή η τεχνική είναι απολύτως έγκυρη χρησιμοποιώντας οποιαδήποτε άλλη αχρησιμοποίητη σημαία.

8.11 Πίνακας ακολουθιών κινουμένων σχεδίων

Τα κινούμενα σχέδια αποτελούνται συνήθως από ζυγό αριθμό καρέ, αν και αυτός δεν είναι αυστηρός κανόνας. Σκεφτείτε για παράδειγμα ένα απλό animation χαρακτήρα με μόνο δύο καρέ: τα πόδια ανοιχτά και κλειστά. Αυτά είναι δύο καρέ. Σκεφτείτε τώρα ένα ενισχυμένο animation, με μια ενδιάμεση φάση κίνησης. Αυτό σημαίνει ότι δημιουργείται η ακολουθία: κλειστά-ενδιάμεσα-ανοιξτε-ανοιξτε-ενδιάμεσα-και ξεκινήστε ξανά. Όπως μπορείτε να δείτε, είναι ένας ζυγός αριθμός, είναι 4.

Οι ακολουθίες κινουμένων σχεδίων 8BP είναι λίστες με 8 καρέ, δεν μπορούν να έχουν περισσότερα, αν και μπορείτε πάντα να κάνετε μικρότερες ακολουθίες.

Τα καρέ μιας ακολουθίας κινουμένων σχεδίων είναι οι διευθύνσεις μνήμης στις οποίες συγκεντρώνονται οι εικόνες από τις οποίες αποτελούνται, και μπορεί να έχουν διαφορετικό μέγεθος, αν και συνήθως είναι το ίδιο. Αν στη μέση της ακολουθίας εισάγετε ένα μηδέν, αυτό σημαίνει ότι η ακολουθία έχει τελειώσει.

Αν και πριν από την V33 υπήρχε μια εντολή RSX που ονομαζόταν **|SETUPSQ** για τη δημιουργία ακολουθιών από τη BASIC, την αφαίρεσα από την V33, επειδή η χρήση της είναι πιο πολύπλοκη από τον ορισμό ακολουθιών από το αρχείο sequences.asm και, στην πραγματικότητα, δεν έχω χρησιμοποιήσει ποτέ την εντολή **|SETUPSQ** σε κανένα από τα παιχνίδια μου, οπότε αποφάσισα να τη θυσιάσω για να εξοικονομήσω μνήμη.

Ας δούμε ένα παράδειγμα δημιουργίας μιας ακολουθίας στο αρχείο sequences.asm.

```
dw MONTOYA_R0,MONTOYA_R1,MONTOYA_R2,MONTOYA_R1,0,0,0,0,0,0,0,0
```

Πριν σας εξηγήσω πώς να αντιστοιχίσετε μια ακολουθία σε ένα sprite, επιτρέψτε μου να σας υπενθυμίσω πώς να αντιστοιχίσετε εικόνες σε sprites: Από την έκδοση V26 της 8BP, υπάρχει η δυνατότητα να συμπεριλάβετε μια λίστα εικόνων (τις ετικέτες τους) σε μια λίστα που ονομάζεται IMAGE_LIST στο αρχείο images_your_game.asm. Με αυτό μπορείτε να αναφέρετε τις εικόνες από τη BASIC με έναν δείκτη αντί για μια διεύθυνση μνήμης. Με αυτόν τον τρόπο δεν χρειάζεται να αναζητάτε διεύθυνσεις μνήμης κάθε φορά που συναρμολογείτε. Αυτό ισχύει για την εντολή:

|SETUPSP, #, 9, <διεύθυνση>.

Το παράδειγμα δείχνει μια ακολουθία κινουμένων σχεδίων με 3 διαφορετικά καρέ, αλλά για να γίνει ομαλή πριν ξεκινήσετε ξανά πρέπει να περάσετε ξανά από το "μεσαίο" καρέ (σημειώστε ότι το δεύτερο και το τέταρτο καρέ είναι τα ίδια), οπότε στο τέλος υπάρχουν 4 καρέ:



Eik. 55 Ακολουθία κινούμενων σχεδίων

Αν θέλατε να δημιουργήσετε μια ακολουθία με περισσότερα από 8 καρέ, θα μπορούσατε απλώς να συνδέσετε δύο ακολουθίες στη σειρά και όταν ο χαρακτήρας φτάσει στο τελευταίο καρέ της πρώτης ακολουθίας να χρησιμοποιήσετε την εντολή **|SETUPSP** για να του αναθέσετε τη δεύτερη ακολουθία.

Οι ακολουθίες κινούμενων σχεδίων συγκεντρώνονται από τη διεύθυνση 33600 και μπορείτε να ορίσετε έως και 31 ακολουθίες κινούμενων σχεδίων (από τον αριθμό 32 και μετά δεν θεωρούνται ακολουθίες, αλλά "ακολουθίες μακροεντολών", που είναι μια άλλη έννοια). Κάθε ακολουθία θα αναγνωρίζεται από έναν αριθμό στο εύρος [1..31]. Η ακολουθία μηδέν δεν υπάρχει, **χρησιμοποιείται για να δηλώσει ότι ένα sprite δεν έχει**

καμία ακολουθία.

Για να αναθέσετε μια ακολουθία σε ένα Sprite χρησιμοποιήστε την εντολή SETUPSP με την παράμετρο 7:

|SETUPSP, <sprite_id>, 7, <αριθμός ακολουθίας>

Με την εντολή αυτή, η ακολουθία κινούμενων σχεδίων εκχωρείται στο sprite στο αντίστοιχο πεδίο του πίνακα sprite και ένα μηδέν τοποθετείται στο πεδίο ID πλαισίου. Επιπλέον, η αντίστοιχη εικόνα εκχωρείται στην πρώτη εικόνα της ακολουθίας. Εάν χρησιμοποιείτε την εντολή |ANIMALL πριν την εκτύπωση ή την εντολή |PRINTSPALL με σημαία animation, ακόμα και αν η SETUPSP τοποθετήσει το animation στο καρέ μηδέν, θα μεταβείτε στο καρέ 1 πριν την εκτύπωση. Αυτό κανονικά δεν αποτελεί πρόβλημα, αλλά στην περίπτωση μιας "ακολουθίας θανάτου" (περισσότερα γι' αυτό αργότερα) όπου για παράδειγμα το πρώτο καρέ είναι για να διαγράψετε το sprite, μπορεί να μην θέλετε να μεταπηδήσετε απευθείας στο καρέ 1.

1. Στην περίπτωση αυτή, ένα απλό τέχνασμα μπορεί να είναι η επανάληψη του καρέ μηδέν στον ορισμό της ακολουθίας θανάτου. Με αυτόν τον τρόπο εξασφαλίζετε ότι το καρέ είναι ορατό. Μια άλλη επιλογή είναι να αφαιρέσετε τη σημαία κίνησης και να την κινήσετε με |ANIMASP μετά την εκτύπωση.

Κάθε ακολουθία αποθηκεύει 8 διευθύνσεις μνήμης που αντιστοιχούν στα 8 πλαίσια, δηλαδή 16 bytes που καταναλώνονται από κάθε ακολουθία.

Το αρχείο ακολουθίας κινούμενων σχεδίων μπορεί να μοιάζει κάπως έτσι:

```
;=====
;-----  
;έως και 31 ακολουθίες κινούμενων σχεδίων  
;=====  
;-----  
;πρέπει να είναι σταθερός πίνακας και όχι μεταβλητός πίνακας  
; ; κάθε ακολουθία περιέχει τις διευθύνσεις των κυκλικών πλαισίων  
;κινούμενων σχεδίων  
;-----  
;Κάθε ακολουθία αποτελείται από 8 διευθύνσεις μνήμης εικόνας.  
;ζυγός αριθμός επειδή οι κινήσεις είναι συνήθως ζυγός αριθμός  
;ένα μηδέν σημαίνει τέλος της ακολουθίας, αν και χρησιμοποιούνται πάντα  
;8 λέξεις.  
;-----  
; by sequence  
;Όταν βρεθεί μηδέν, γίνεται νέα αρχή.  
;εάν δεν υπάρχει μηδέν, μετά το πλαίσιο 8 αρχίζει πάλι.  
;Εάν σε ένα Sprite ανατεθεί η ακολουθία μηδέν, δεν έχει καμία ακολουθία.  
;Ξεκινάμε από την ακολουθία 1  
;----- ακολουθίες animation χαρακτήρων montoya -----  
_SEQUENCES_LIST  
dw MONTOYA_R0,MONTOYA_R1,MONTOYA_R2,MONTOYA_R1,0,0,0,0,0,0,0,0 ;1  
dw MONTOYA_U0,MONTOYA_U1,MONTOYA_U0,MONTOYA_U2,0,0,0,0,0,0 ;2  
dw MONTOYA_U0,MONTOYA_U1,MONTOYA_U0,MONTOYA_U2,0,0,0,0,0,0 ;3  
dw MONTOYA_UL0,MONTOYA_UL1,MONTOYA_UL2,MONTOYA_UL1,0,0,0,0,0,0 ;4  
dw MONTOYA_L0,MONTOYA_L1,MONTOYA_L2,MONTOYA_L1,0,0,0,0,0,0 ;5  
dw MONTOYA_DL0,MONTOYA_DL1,MONTOYA_DL2,MONTOYA_DL1,0,0,0,0,0,0 ;6  
dw MONTOYA_D0,MONTOYA_D1,MONTOYA_D0,MONTOYA_D2,0,0,0,0,0,0,0 ;7  
dw MONTOYA_DR0,MONTOYA_DR1,MONTOYA_DR2,MONTOYA_DR1,0,0,0,0,0,0 ;8  
;----- soldier animation sequences ----- dw  
SOLDIER_R0,SOLDIER_R2,SOLDIER_R1,SOLDIER_R2,0,0,0,0,0,0 ;9  
dw SOLDIER_L0,SOLDIER_L2,SOLDIER_L1,SOLDIER_L2,0,0,0,0,0,0 ; 10
```

8.12 Ειδικές ακολουθίες κινούμενων σχεδίων

Διάφοροι τύποι ακολουθιών κινούμενης εικόνας είναι διαθέσιμοι στην 8BP:

Τύπος ακολουθίας	περιγραφή
Κανονική ακολουθία	Η ακολουθία των καρέ κινουμένων σχεδίων επαναλαμβάνεται ξανά και ξανά. Τελειώνουν είτε σε μια κατεύθυνση εικόνας είτε σε μια
	ένα μηδέν αν θέλουμε να δημιουργήσουμε μια ακολουθία με λιγότερες από 8 εικόνες
Ακολουθία θανάτου	Το τελευταίο καρέ της ακολουθίας είναι ένα "1". Αυτό λέει στην 8BP να αλλάξει την κατάσταση του Sprite σε μηδέν. Κανονικά το καρέ πριν από το "1" είναι μια εικόνα διαγραφής.
Τέλος ακολουθίας	Μετά την εκτέλεση της ακολουθίας, το Sprite εξαντλεί το animation. Το τελευταίο καρέ της ακολουθίας είναι ένα "2". Αυτό λέει στην 8BP να αφαιρέσει τη σημαία κίνησης από την κατάσταση Sprite.
Αλυσίδα ακολουθίας	Μετά τη διέλευση της ακολουθίας, το τελευταίο πλαίσιο υποδεικνύει το αναγνωριστικό της επόμενης ακολουθίας που θα ανατεθεί στο Sprite. Χρησιμοποιώντας αυτόν τον τύπο ακολουθιών μπορείτε να δημιουργήσετε ακολουθίες κινουμένων σχεδίων μεγαλύτερες από 8 καρέ.
μακρο-ακολουθίες	Επιτρέπουν τη σύνδεση μιας ακολουθίας ανάλογα με την ταχύτητα του Sprite, αυτόματα.

8.12.1 Ακολουθίες θανάτου

Η βιβλιοθήκη 8BP σας επιτρέπει να κάνετε "ακολουθίες θανάτου", οι οποίες είναι ακολουθίες όπου το sprite περνάει σε ανενεργή κατάσταση στο τέλος της ακολουθίας. Αυτό υποδεικνύεται από ένα απλό "1" ως τιμή της διεύθυνσης μνήμης του τελευταίου καρέ. Αυτές οι ακολουθίες είναι πολύ χρήσιμες για τον ορισμό εκρήξεων εχθρών που κινούνται με |ANIMA ή

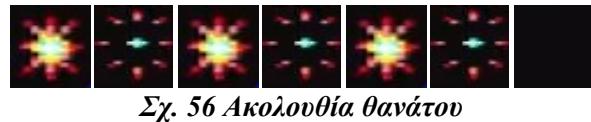
|ANIMALL. Αφού τους χτυπήσετε με τη βολή σας, μπορείτε να τους συνδέσετε μια ακολουθία κινουμένων σχεδίων θανάτου και στους επόμενους κύκλους του παιχνιδιού θα περάσουν από τις διάφορες φάσεις κινουμένων σχεδίων της έκρηξης, και όταν φτάσουν στην τελευταία θα περάσουν σε ανενεργή κατάσταση, χωρίς να εκτυπώνονται πλέον. Αυτή η ανενεργή κατάσταση γίνεται αυτόματα, οπότε αυτό που έχετε να κάνετε είναι απλά να ελέγξετε τη σύγκρουση της βολής σας με τους εχθρούς και αν συγκρουστεί με κάποιον από αυτούς να αλλάξετε την κατάσταση με το SETUPSP ώστε να μην μπορεί να συγκρουστεί πλέον και να του αναθέσετε την ακολουθία animation θανάτου, επίσης με το SETUPSP.

Αν χρησιμοποιήσετε μια ακολουθία θανάτου, μην ξεχάσετε να βεβαιωθείτε ότι το τελευταίο καρέ πριν βρείτε το "1" είναι εντελώς άδειο, ώστε να μην υπάρχει κανένα ίχνος της έκρηξης.

Παράδειγμα ακολουθίας θανάτου (σημειώστε ότι περιλαμβάνει ένα "1"):

dw EXPLOSION 1.EXPLOSION 2.ExPLOSION 3.1,0,0,0,0,0,0,0,0

Ένα ενδιαφέρον εφέ είναι να περνάτε επανειλημμένα από διάφορα καρέ πριν καταλήξετε σε ένα μαύρο καρέ που χρησιμεύει για να διαγράψετε



Σχ. 56 Ακολουθία θανάτου

Το "1" εμφανίζεται τώρα στην όγδοη θέση:

**dw EXPLOSION_1,EXPLOSION_2, EXPLOSION_1,EXPLOSION_2, EXPLOSION_1,EXPLOSION_2,
ExPLOSION_3,1**

Να θυμάστε ότι αν χρησιμοποιήσετε την εντολή `PRINTSPALL` με ενεργή τη σημαία `animation`, πρώτα γίνεται `animation` και μετά εκτυπώνεται, οπότε η πρώτη εικόνα της ακολουθίας θανάτου δεν θα εμφανιστεί. Ένα απλό τέχνασμα για να την κάνετε να εμφανιστεί είναι να την επαναλάβετε δύο φορές.

8.12.2 Ακολουθίες τέλους

Οι αλληλουχίες τέλους υπάρχουν από την έκδοση V42. Επιτρέπουν τη σύνδεση μιας ακολουθίας κίνησης με ένα Sprite που θέλουμε να εκτελεστεί μόνο μία φορά. Στο τέλος της ακολουθίας, το Sprite δεν θα έχει καμία σημαία κίνησης. Οι άλλες σημαίες της κατάστασής του είναι σεβαστές.

Πρέπει απλώς να βάλουμε ένα "2" στο τελευταίο πλαισιο της ακολουθίας.

Ακολουθούν δύο παραδείγματα. Μόλις η ακολουθία φτάσει στο καρέ "2", το sprite δεν κινείται πλέον.

SEQUENCES_LIST
dw
MONTOYA_R0,MONTOYA_R1,MONTOYA_R2,MONTOYA_R1,2,0,0,0,0,
0 dw IMG1, IMG2, IMG3, IMG4, IMG5, IMG6, IMG7,2

8.12.3 Αλυσίδες αλληλουχιών

Οι αλυσιδωτές ακολουθίες υπάρχουν από την έκδοση 8BP V42 . Επιτρέπουν την κατασκευή ακολουθιών μεγαλύτερων των 8 πλαισίων με την αλυσιδωτή σύνδεση ακολουθιών μεταξύ τους.

Ο μηχανισμός είναι απλός. Απλά το τελευταίο καρέ της ακολουθίας πρέπει να είναι ο αριθμός της ακολουθίας που θέλετε να αναθέσετε στην επόμενη.

Όπως μπορείτε να φανταστείτε, δεν θα μπορέσετε να αναθέσετε την ακολουθία "1" ή "2", αφού αυτοί οι αριθμοί σημαίνουν "πεθαίνω" ή "τελειώνω". Δηλαδή, θα είστε σε θέση να αντιστοιχίσετε αλυσιδωτές ακολουθίες ξεκινώντας από τον αριθμό 3.

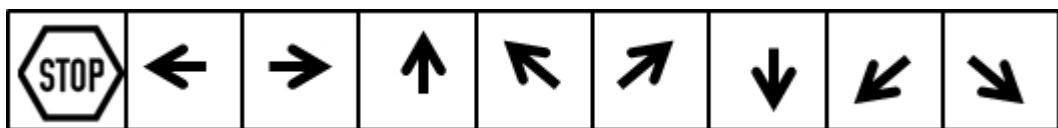
Σε αυτό το παράδειγμα έχω συνδέσει αλυσιδωτά τις ακολουθίες 4 και 5 έτσι ώστε μετά τη μία να ανατίθεται η άλλη και αντίστροφα. Είναι σαν να έχουμε μια ακολουθία 14 καρέ. Θα μπορούσαμε να αλυσοδέσουμε περισσότερες ακολουθίες και να κάνουμε το animation πολύ μεγαλύτερο. Κάθε ακολουθία που προσθέτουμε είναι 7 επιπλέον καρέ (όχι 8) επειδή χρειαζόμαστε το τελευταίο για να υποδείξουμε την επόμενη ακολουθία. Μπορείτε επίσης να τα κάνετε μικρότερα και να τα γεμίσετε με μηδενικά έως και 8 καρέ, αλλά ο αριθμός ακολουθίας πρέπει να εμφανίζεται πριν από κάθε μηδενικό, διότι όταν εμφανίζεται ένα μηδενικό, η κίνηση ανακυκλώνεται.

8.12.4 Μακρο-ακολουθίες κινούμενων σχεδίων

Πρόκειται για μια "προηγμένη" λειτουργία που είναι διαθέσιμη από την έκδοση V25 της βιβλιοθήκης 8BP. Μια "μακροακολουθία" είναι μια ακολουθία που αποτελείται από ακολουθίες. Κάθε μία από τις

που αποτελούν αλληλουχίες κινούμενων σχεδίων είναι το κινούμενο σχέδιο που πρέπει να εκτελεστεί προς μια συγκεκριμένη κατεύθυνση. Η κατεύθυνση καθορίζεται από τα χαρακτηριστικά ταχύτητας του sprite, τα οποία βρίσκονται στον πίνακα sprite. Έτσι, όταν κινούμε ένα sprite με |ANIMALL, θα αλλάξει αυτόματα την ακολουθία κινούμενων σχεδίων χωρίς να χρειαστεί να κάνουμε τίποτα (στην πραγματικότητα δεν χρειάζεται να καλέσετε το |ANIMALL, επειδή το |PRINTSPALL το κάνει ήδη αυτό εσωτερικά αν ορίσετε μια παράμετρο).

Οι ακολουθίες μακροεντολών αριθμούνται ξεκινώντας από το 32. Είναι πολύ σημαντικό να τοποθετήσετε τις ακολουθίες μέσα στην ακολουθία μακροεντολών με τη σωστή σειρά, δηλαδή η πρώτη ακολουθία πρέπει να είναι για όταν ο χαρακτήρας είναι ακίνητος, η επόμενη για όταν ο χαρακτήρας πηγαίνει αριστερά ($Vx < 0$, $Vy = 0$), η επόμενη για δεξιά ($Vx > 0$, $Vy = 0$), κ.λπ. με την ακόλουθη σειρά (προσέξτε γιατί είναι εύκολο να κάνετε λάθος):



Σχ. 57 Σειρά ακολουθιών σε μια μακροακολούθια

Εάν η ακολουθία που έχει εκχωρηθεί στη θέση ηρεμίας είναι μηδέν, τότε απλά κινείται με την τελευταία εκχωρηθείσα ακολουθία.

Οι ακολουθίες μακροεντολών πρέπει να καθορίζονται στο αρχείο sequences_yourgame.asm, ένα παράδειγμα του οποίου δίνεται παρακάτω:

```
=====
;ακολουθίες κινούμενων σχεδίων
=====
_SEQUENCES_LIST
dw NAVE,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
dw
JOE1,JOE2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2
UP JOE dw
JOE7,JOE8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
;3 DW JOE dw
JOE3,JOE4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4 R
JOE dw
JOE5,JOE6,0,0,0,0,0,0,0,0,0,0,0,0,5 L JOE

_MACRO_SEQUENCES
-----MACRO SEQUENCES -----
είναι ομάδες ακολουθιών, μία για κάθε κατεύθυνση:
; ακόμα, αριστερά, δεξιά, πάνω, πάνω-αριστερά, πάνω-δεξιά, κάτω,
κάτω-αριστερά, κάτω-δεξιά
Οι αριθμοί αριθμούνται από το 32 και μετά
db 0,5,4,2,5,4,3,5,4;η ακολουθία 32 περιέχει τις ακολουθίες του στρατιώτη Joe
```

Με αυτόν τον ορισμό ακολουθίας μπορούμε να δημιουργήσουμε ένα απλό παιχνίδι που μας επιτρέπει να μετακινούμε τον "Joe" στην οθόνη χωρίς να ελέγχουμε την ακολουθία των κινήσεών του. Αναθέτουμε την ακολουθία 32 και μεταβάλλοντας την ταχύτητα, η εντολή |ANIMA (που καλείται από το εσωτερικό του |PRINTSPALL) είναι υπεύθυνη για την αλλαγή της ακολουθίας κινούμενων σχεδίων, εάν η ταχύτητά της υποδηλώνει αλλαγή κατεύθυνσης. Για να μετακινήσουμε το sprite

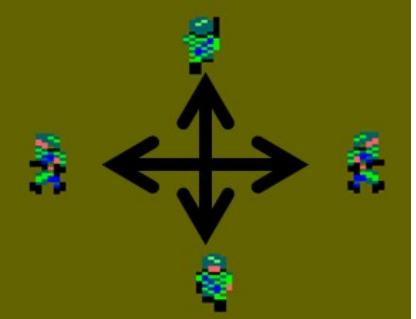
πρέπει να επικαλεστούμε το |**AUTOALL**, αφού το πάτημα των χειριστηρίων δεν αλλάζει τις συντεταγμένες του αλλά την ταχύτητά του και το |**AUTOALL** θα ενημερώσει τις συντεταγμένες του sprite ανάλογα με την ταχύτητά του.

10 MNMH 24999	
20 MODE 0:INK 0,12	
30 ΣΤΟ ΔΙΑΛΕΙΜΜΑ GOSUB 280	
40 CALL &6B78	
50 DEFINT a-z	
111 x=36:y=100	
120 SETUPSP,31,0,0,0,&X1111	
130 SETUPSP,31,7,2: SETUPSP,31,7,32	

```

140 |LOCATESP,31,y,x
160 |SETLIMITS,0,80,0,200
161 |PRINTSPALL,0,1,0
190 'αρχίζει κύκλο παιχνιδιού
199 vy=0:vx=0
200 AN INKEY(27)=0 TOTE vx=1: GOTO 220
210 EAN INKEY(34)=0 TOTE vx=-1
220 AN INKEY(69)=0 TOTE vy=2: GOTO 240
230 AN INKEY(67)=0 TOTE vy=-2
240 |SETUPSP,31,5,vy,vx
250 |AUTOALL:|PRINTSPALL
270 GOTO 199
280 |MUSIC:MODE 1: INK 0,0:PEN 1

```



Σημειώστε ότι δεν έχω ορίσει την ακολουθία για το πότε ο χαρακτήρας δεν κινείται. Σε αυτή τη θέση έχω βάλει ένα μηδέν στην ακολουθία μακροεντολών. Αυτό σημαίνει ότι, αν ο χαρακτήρας ξεκινήσει ακίνητος, δεν ξέρετε ποια ακολουθία να ορίσετε, καθώς δεν χρησιμοποιείται "τελευταία" ακολουθία. Γι' αυτό εκχωρώ την ακολουθία 2 πριν εκχωρήσω την 32, ώστε να βεβαιωθώ ότι ο χαρακτήρας έχει ήδη μια ακολουθία, ακόμη και αν στέκεται ακίνητος.

130 SETUPSP, 31, 7, 7, 7, 2: SETUPSP, 31, 7, 32

9 Το πρώτο σας απλό παιχνίδι

Τώρα έχετε τις γνώσεις για να δοκιμάσετε ένα πρώτο βήμα στη δημιουργία βιντεοπαιχνιδιών. Για να το κάνετε αυτό, ας δούμε ένα απλό παράδειγμα ενός στρατιώτη που θα ελέγχετε, κάνοντάς τον να περπατάει δεξιά και αριστερά στην οθόνη.

Ας υποθέσουμε ότι έχουμε επεξεργαστεί έναν στρατιώτη, χάρη στο SPEDIT. Και έχουμε κατασκευάσει τις ακολουθίες κίνησής του, οι οποίες έχουν οριστεί στο αρχείο **"sequences_mygame.asm"** και έχουν αφεθεί με το αναγνωριστικό 9 και 10 για τις αριστερές και δεξιές κατευθύνσεις κίνησης αντίστοιχα.

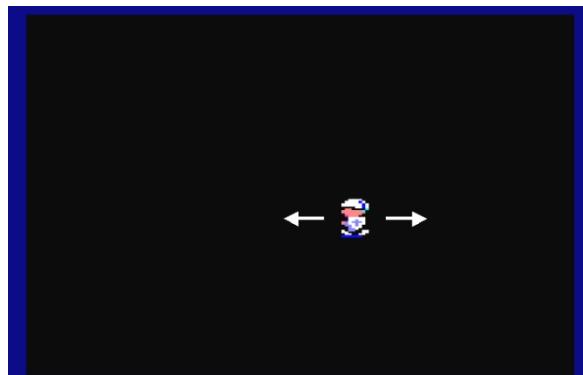
Οι δύο ακολουθίες κινούμενων σχεδίων έχουν δημιουργηθεί από το αρχείο sequences.asm.

```
10 MNHMH 24499
20 MODE 0: DEFINT A-Z: CALL &6B78:' install RSX
25 call &bc02:'restore default palette just in case'.
26 ink 0,0:'μαύρο φόντο
30 FOR j=0 TO 31:|SETUPSP,j,0,&X0:NEXT:'επαναφορά των sprites
40 |SETLIMITS,12,80,0,186: ' ορίστε τα όρια της οθόνης του παιχνιδιού
50 x=40:y=100:' συντεταγμένες χαρακτήρα 51|SETUPSP,0,0,1:' κατάσταση χαρακτήρα 52|SETUPSP,0,7,9:'ακολουθία κίνησης που έχει οριστεί για την έναρξη
53|LOCATESP,0,y,x:'τοποθέτηση του sprite (χωρίς να το εκτυπώσουμε ακόμα)

60 'κύκλος παιχνιδιού
70 gosub 100
80 |PRINTSPALL,0,0
90 goto 60

99 ' ρουτίνα κίνησης χαρακτήρα -----
100 AN INKEY(27)=0 TOTE AN dir<>>0 TOTE |SETUPSP,0,7,9:dir=0:return ELSE
|ANIMA,0:x=x+1:GOTO 120
110 AN INKEY(34)=0 TOTE AN dir<>>1 TOTE |SETUPSP,0,7,10:dir=1:return ELSE
|ANIMA,0:x=x-1
120 |LOCATESP,0,y,x
130 ΕΠΙΣΤΡΟΦΗ
```

Με αυτή τη λίστα έχετε ήδη ένα μίνι-παιχνίδι που σας επιτρέπει να ελέγχετε έναν στρατιώτη και να τον κάνετε να τρέχει οριζόντια. Σημειώστε ότι, αν όταν περπατάτε προς τα αριστερά υπερβείτε την ελάχιστη τιμή του ορίου που έχει οριστεί με την εντολή |SETLIMITS, ο χαρακτήρας θα "ψαλιδιστεί", δείχνοντας μόνο το μέρος που βρίσκεται μέσα στην επιτρεπόμενη περιοχή παιχνιδιού.



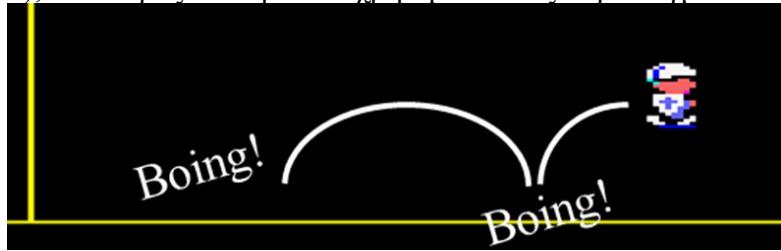
Εικ. 58 Ένα απλό παιχνίδι

9.1 Τώρα, ας πηδήξουμε! Boing, boing!

Στο παραπάνω παράδειγμα ο χαρακτήρας μας κινείται μόνο από αριστερά προς τα

δεξιά. Αν θέλουμε να προγραμματίσουμε ένα áλμα, μπορούμε να το κάνουμε αποθηκεύοντας την κάθετη τροχιά σε μια

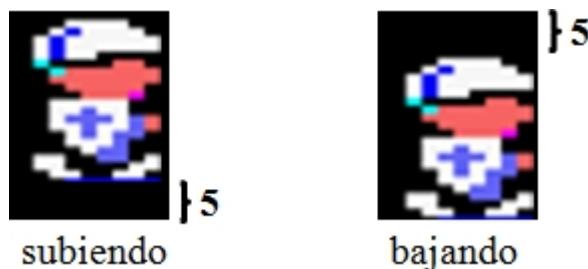
Συστοιχία BASIC. Αργότερα θα δούμε έναν καλύτερο τρόπο για να το κάνουμε (με διαδρομές 8BP), αλλά προς το παρόν θα χρησιμεύσει ως παράδειγμα.



Εικ. 59 ο χαρακτήρας μας μπορεί να πηδήξει

Η τροχιά άλματος ορίζεται για τη συντεταγμένη Y. Πρώτα ανεβαίνει 5 γραμμές ταυτόχρονα, μετά 4, μετά 3 κ.λπ. μέχρι να φτάσει στο μηδέν. Σε εκείνο το σημείο η κατεύθυνση της κίνησης αντιστρέφεται και αρχίζουμε να κατεβαίνουμε 1 γραμμή, μετά 2, μετά 3, κ.λπ. μέχρι την 5.

Για να μην αφήνει ίχνη όταν το ομοίωμα ανεβαίνει και κατεβαίνει, θα πρέπει να έχουμε ένα σχέδιο του ομοιώματος που ανεβαίνει με 5 μαύρες γραμμές από κάτω για να σβήνει τον εαυτό του όταν ανεβαίνει και με τον ίδιο τρόπο, μια άλλη εικόνα με 5 μαύρες γραμμές από πάνω για να κατεβαίνει. Σε αυτή την περίπτωση είναι οι εικόνες 22 και 23 για να πηδήξει προς τα δεξιά και 24,25 προς τα αριστερά.



Εικ. 60 Εικόνα προς τα πάνω και προς τα κάτω

Στο ζενίθιο σημείο του άλματος, η εικόνα προς τα πάνω πρέπει να αλλάξει σε εικόνα προς τα κάτω, αλλά πρώτα πρέπει να ανέβουμε 5 γραμμές ταυτόχρονα, γιατί αν συγκρίνετε τις δύο εικόνες, θα συνειδητοποιήσετε ότι αν πάτε από τη μία στην άλλη απευθείας, είναι σαν να κατεβαίνετε 5 γραμμές.

```

10 MNHMH 24999
20 MODE 0: DEFINT A-Z: CALL &6B78:' install RSX
25 ON BREAK GOSUB 2800
30 CALL &BC02:'επαναφορά της προεπιλεγμένης παλέτας για παν
ενδεχόμενο'.
40 INK 0,0: "μαύρο φόντο".
50 FOR j=0 ΤΟ 31:|SETUPSP,j,0,&X0:NEXT:'επαναφορά των sprites
80 |SETLIMITS,12,80,0,186: ' ορίζουμε τα όρια της οθόνης
90 x=40:y=100:' συντεταγμένες του χαρακτήρα
100 |SETUPSP,0,0,1:' κατάσταση χαρακτήρα
110 |SETUPSP,0,7,9:'Ακολουθία κινούμενων σχεδίων που εκχωρείται στην
αρχή
120 |LOCATESP,0,y,x:'Τοποθετήστε το sprite (χωρίς να το εκτυπώσετε
ακόμα)
121 DIM jump(24):' δεδομένα άλματος
122 ΟΙΚΟΝΕΔΟ:1:k=50:ΣΧΕΙΛΑΣΜΟΣ k=640:ισο:ΟΙΚΟΝΕΧΟ
skip92,150:ΣΧΕΙΛΑΣΜΟΣ 92,400:'δάπεδο και τοίχος
126 |MUSIC,0,0,5: 'Η μουσική αρχίζει να παιζει
130 ----- "κύκλος παιχνιδιού".

```

150 |LOCATESP,0,y,x:|PRINTSPALL,0,0
GOSUB 170
160 GOTO 130:' τέλος του κύκλου παιχνιδιού

170 ρουτίνα μετακίνησης χαρακτήρων -----

171 AN jump =0 TOTE AN INKEY(67)=0 TOTE jump=1:|SETUPSP,0,9,DIR*2+22

180 AN INKEY(27)=0 TOTE x=x+1:av jump=0 τότε AN dir<>0 TOTE |SETUPSP,0,7,9:dir=0:x=x-1:RETURN ELSE |ANIMA,0:GOTO 210

190 AN INKEY(34)=0 TOTE x=x-1:av jump=0 τότε AN dir<>1 TOTE |SETUPSP,0,7,9:dir=1:x=x+1:RETURN ELSE |ANIMA,0

260 ρουτίνα άλματος -----

**270 EAN jump=11 TOTE |SETUPSP,0,9,DIR*2+23 ELSE EAN jump=23
TOTE**

y=y+jump(jump):jump=0:|SETUPSP,0,7,DIR+9:return

**280 y=y+jump(jump)
jump=jump+1**

310 επιστροφή

2800 |MUSIC:MODE 1: INK 0,0:PEN 1

Όπως μπορείτε να δείτε στη λίστα, αν πατήσετε το πλήκτρο "Q", η μεταβλητή "jump" είναι ίση με 1 και εκείνη τη στιγμή η λογική του ομοιώματος περιπλέκεται επειδή απαιτείται η εκτέλεση μιας εντολής IF για να αλλάξει η εικόνα όταν φτάσει στο σημείο ζενίθ και είναι επίσης απαραίτητο να ενημερωθεί η συντεταγμένη Y του ομοιώματος και η μεταβλητή "jump".

Αργότερα θα δούμε πώς να το κάνουμε αυτό με μια πιο προηγμένη τεχνική, χρησιμοποιώντας "διαδρομές" sprite. Οι προγραμματιζόμενες διαδρομές της 8BP παρέχουν μια πιο αποτελεσματική μέθοδο για να κάνετε κάτι τέτοιο, οπότε θα δείτε τον χαρακτήρα σας να πηδάει πολύ πιο γρήγορα. Οι διαδρομές θα σας επιτρέψουν να εκτελέσετε μια τροχιά (ένα άλμα, έναν κύκλο κ.λπ.) χωρίς να χρειάζεται να ελέγχετε τις συντεταγμένες κάθε στιγμή. Και μπορείτε επίσης να αλλάξετε την κατάσταση ενός sprite στη μέση διαδρομής, ή να αλλάξετε τη σχετική του εικόνα, την ακολουθία του ή ακόμα και να αλλάξετε τη διαδρομή του, συνδέοντας διαφορετικές διαδρομές.

10 Σετ οθόνης: διάταξη ή "χάρτης πλακιδίων".

10.1 Ορισμός και χρήση διάταξης

Συχνά θα θέλετε τα παιχνίδια σας να αποτελούνται από ένα σύνολο οθονών όπου ο χαρακτήρας πρέπει να συλλέξει θησαυρούς ή να αποφύγει τους εχθρούς σε έναν λαβύρινθο. Σε τέτοιες περιπτώσεις καθίσταται απαραίτητη η χρήση ενός πίνακα όπου ορίζετε τα συστατικά μπλοκ κάθε "λαβύρινθου" ή της λεγόμενης "διάταξης" της οθόνης. Μερικές φορές αυτή η έννοια ονομάζεται επίσης "χάρτης πλακιδίων" (το "tile" είναι η αγγλική λέξη για το "πλακίδιο").

Στη βιβλιοθήκη **8BP** έχετε έναν απλό μηχανισμό για να το κάνετε αυτό, ο οποίος παρέχει επίσης μια λειτουργία σύγκρουσης για να ελέγχετε αν ο χαρακτήρας σας έχει μετακινηθεί σε μια περιοχή που καταλαμβάνεται από ένα "τούβλο". Αυτός ο μηχανισμός ονομάζεται "διάταξη". Στην 8BP μια διάταξη ορίζεται από έναν πίνακα 20x25 "μπλοκ" των 8x8 pixels, τα οποία μπορεί να είναι κατειλημμένα ή όχι. Δηλαδή, υπάρχουν τόσα μπλοκ όσα και οι χαρακτήρες στην οθόνη στη λειτουργία 0.

Για να εκτυπώσετε μια διάταξη στην οθόνη έχετε την εντολή:

```
| LAYOUT, <y>, <x>, @string$, <string$, <string$, <string$, <string$,
<string$, <string$, <string$.
```

Αυτή η ρουτίνα εκτυπώνει μια σειρά από sprites για τη δημιουργία της διάταξης ή του "λαβύρινθου" κάθε οθόνης. Ο πίνακας ή ο "χάρτης διάταξης" αποθηκεύεται σε μια περιοχή μνήμης που χειρίζεται 8BP, έτσι ώστε όταν εκτυπώνετε μπλοκ δεν εκτυπώνετε μόνο στην οθόνη, αλλά γεμίζετε και την περιοχή μνήμης που καταλαμβάνει η διάταξη (20x25 bytes), όπου κάθε byte αντιπροσωπεύει ένα μπλοκ.

Οι συντεταγμένες y,x περνούν σε μορφή χαρακτήρων,
δηλαδή το y παίρνει τιμές [0,24].
Το x παίρνει τιμές [0,19].

Τα μπλοκ που εκτυπώνονται από τη συνάρτηση **|LAYOUT** κατασκευάζονται με συμβολοσειρές χαρακτήρων και κάθε χαρακτήρας αντιστοιχεί σε ένα sprite που πρέπει να υπάρχει. Έτσι, το μπλοκ "Z" αντιστοιχεί στην εικόνα που έχει ανατεθεί στο sprite 31. Το μπλοκ "Y" αντιστοιχεί στην εικόνα που έχει ανατεθεί στο sprite 30, και ούτω καθεξής.

Τα sprites που πρόκειται να εκτυπωθούν ορίζονται με μια συμβολοσειρά, οι χαρακτήρες της οποίας (32 δυνατοί) αντιπροσωπεύουν ένα από τα sprites που ακολουθούν αυτόν τον απλό κανόνα, όπου η μόνη εξαίρεση είναι το κενό διάστημα που αντιπροσωπεύει την απουσία ενός sprite.

Χαρακτήρας	Id Sprite	Κωδικός ASCII
" "	KANENAΣ	
";"	0	59
"<"	1	
"="		
">"		
"?"		63
"@"	5	
"A"		65
"B"		
"C"	8	67
"D"		
"E"	10	69
"F"		70
"G"		71
"H"		
"I"		
"J"		
"K"		75
"L"		
"M"		
"N"		78
"O"		79
"P"	21	80
"Q"		81
"R"	23	82
"S"		
"T"	25	84
"U"	26	85
"V"		86
"W"		87
"X"	29	88
"Y"	30	
"Z"	31	90

Πίνακας 4 Αντιστοίχιση χαρακτήρων και Sprite για την εντολή |AYOUT

Η @string είναι μια μεταβλητή συμβολοσειράς. Δεν μπορείτε να περάσετε το αλφαριθμητικό απευθείας. Δηλαδή, θα ήταν παράνομο να κάνετε κάτι τέτοιο:

|AYOUT, 1, 0, "ZZZ YYY".

Το σωστό είναι:

String\$ = "ZZZ YYY".

|AYOUT, 1, 0, @string\$

Προσέξτε η συμβολοσειρά να μην είναι κενή, διαφορετικά ο υπολογιστής μπορεί να καταρρεύσει! Επιπλέον, πρέπει να προτάξετε τη μεταβλητή συμβολοσειρά με το σύμβολο "@" για να

ότι η βιβλιοθήκη μπορεί να μεταβεί στη διεύθυνση μνήμης όπου είναι αποθηκευμένο το αλφαριθμητικό και να το διασχίσει, εκτυπώνοντας ένα προς ένα τα αντίστοιχα sprites.

Θα πρέπει να σημειώσετε ότι τα κενά σημαίνουν ότι δεν υπάρχει sprite, δηλαδή τίποτα δεν εκτυπώνεται στις θέσεις που αντιστοιχούν στα κενά. Εάν υπήρχε προηγουμένως κάτι σε αυτή τη θέση, δεν θα διαγραφεί. Αν θέλετε να διαγράψετε, πρέπει να ορίσετε ένα sprite διαγραφής 8x8, όπου όλα είναι μηδενικά.

Αν και χρησιμοποιείτε τα sprites για να εκτυπώσετε τη διάταξη, αμέσως μετά την εκτύπωση μπορείτε να επαναπροσδιορίσετε τα sprites με το |SETUPSP και να τους αναθέσετε εικόνες στρατιωτών, τεράτων ή οτιδήποτε άλλο θέλετε, δηλαδή η διάταξη "βασίζεται" στο μηχανισμό sprite για να εκτυπώσει, αλλά δεν περιορίζει τον αριθμό των sprites, επειδή έχετε και τα 32 sprites να είναι ό,τι θέλετε αμέσως μετά την εκτύπωση της διάταξης.

Για την ανίχνευση συγκρούσεων με τη διάταξη έχετε τη συνάρτηση COLAY, η οποία μπορεί να χρησιμοποιηθεί με μεταβλητό αριθμό παραμέτρων.

| COLAY,<κατώφλι ASCII>, @collision , <αριθμός sprite>.
| COLAY, @collision , <αριθμός στίξης>
| COLAY, <αριθμός αρίθμησης>, <αριθμός αρίθμησης>.
| COLAY

Δεδομένου ενός sprite και ανάλογα με τις συντεταγμένες και το μέγεθός του, αυτή η συνάρτηση θα βρει αν συγκρούεται με τη διάταξη και θα σας προειδοποιήσει μέσω της μεταβλητής σύγκρουσης, η οποία πρέπει να έχει οριστεί προηγουμένως.

Η παράμετρος <κατώφλι ASCII> είναι προαιρετική και χρησιμοποιείται έτσι ώστε η εντολή να μην θεωρεί κωδικούς ASCII κάτω από αυτό το κατώφλι ως συγκρούσεις. Από προεπιλογή είναι 32 (που αντιστοιχεί στο λευκό διάστημα). Για να το καταλάβετε αυτό, είναι απαραίτητο να λάβετε υπόψη την αντιστοιχία μεταξύ των τιμών ASCII και των Sprites που φαίνεται στον προηγούμενο πίνακα. Για παράδειγμα, αν θέσουμε το κατώφλι σε 69 ("E" κωδικός, sprite 10), τότε τα sprites 9, 8, 7, 6, 5, 4, 3, 2, 1 και 0 δεν θα είναι "collisible", οπότε αν ο χαρακτήρας μας περάσει από πάνω τους, η σύγκρουση απλώς δεν θα ανιχνευθεί.

Η κλήση του COLAY με την παράμετρο threshold είναι απαραίτητη μόνο φορά, καθώς οι επόμενες κλήσεις λαμβάνουν ήδη υπόψη αυτό το threshold.

Παράδειγμα χρήσης:

col%=0

| COLAY, @col%, 20: REM αντό είναι ένα παράδειγμα με spriteID=20

Εάν καλέσετε την εντολή COLAY χωρίς παραμέτρους, θα θεωρήσει τις τελευταίες που χρησιμοποιήθηκαν. Με αυτόν τον τρόπο μπορείτε να γλιτώσετε το πέρασμα παραμέτρων και να επιταχύνετε την εντολή κατά 0,5 ms.

Εάν δεν υπάρχει σύγκρουση, η μεταβλητή θα πάρει την τιμή μηδέν. Η σύγκρουση συμβαίνει αν το sprite συγκρουστεί με οποιοδήποτε στοιχείο διάταξης εκτός από το κενό (" "), του οποίου ο κωδικός ASCII είναι 32. Αν χρησιμοποιείται το κατώφλι, η σύγκρουση συμβαίνει αν το στοιχείο διάταξης έχει ASCII μεγαλύτερο από το κατώφλι

πον ορίζετε.

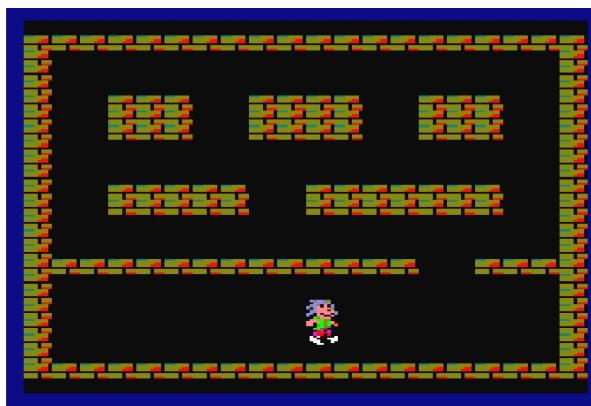
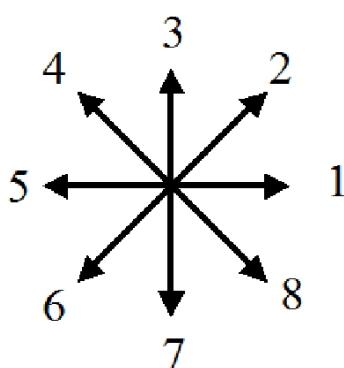
Θα δούμε ένα παράδειγμα δημιουργίας μιας διάταξης και μετακίνησης ενός χαρακτήρα μέσα στη διάταξη, διορθώνοντας τη θέση του αν έχει συγκρουστεί.

Δύο τελευταίες παρατηρήσεις σχετικά με την εντολή |COLAY|:

- Η εντολή **|COLAY** δεν επηρεάζεται από τη ρύθμιση εναισθησίας σύγκρουσης sprite (διαμορφώνεται με **|COLSP,34, dy, dx**). Η ρύθμιση εναισθησίας σύγκρουσης επηρεάζει μόνο τις εντολές **|COLSP** και **|COLSPALL**.
 - Η εντολή **|COLAY** δεν λαμβάνει υπόψη το πραγματικό μέγεθος των εικόνων που χρησιμοποιούνται ως "Πλακάκια" ή "μπλοκ" της διάταξης. Δηλαδή, θεωρεί ότι όλα τα μπλοκ που καθορίζονται στην εντολή **|LAYOUT** είναι 8x8 pixels της λειτουργίας 0 (4 bytes x 8 γραμμές), ακόμα και αν τοποθετήσετε μεγαλύτερες εικόνες.

10.2 Παράδειγμα παιχνιδιού με διάταξη

Θα εξελίξουμε λίγο το gameplay που παρουσιάστηκε στο προηγούμενο κεφάλαιο, όσον αφορά τον έλεγχο του χαρακτήρα. Αυτή τη φορά θα χρησιμοποιήσουμε ως παράδειγμα τον Montoya, ο οποίος έχει 8 αλληλουχίες animation, η κάθε μία για να κινείται προς διαφορετική κατεύθυνση. Στις ακολουθίες κινουμένων σχεδίων έχει δοθεί ένας αριθμός από το 1 έως το 8.



Εικ. 61 Χρήση της διάταξης σε ένα παιγνίδι

Στη ρουτίνα ελέγχου των χαρακτήρων έχουμε συμπεριλάβει τη σύγκρουση με τη διάταξη. Ανάλογα με την κατεύθυνση προς την οποία κινούμαστε, τροποποιούμε τις "νέες" συντεταγμένες (yp , xn) και καλούμε τη συνάρτηση σύγκρουσης με τη διάταξη |COLAY,0 για να ελέγξουμε αν το sprite 0 (ο χαρακτήρας μας) έχει συγκρουστεί. Αν έχει συγκρουστεί, διορθώνουμε τις συντεταγμένες (τη μία ή και τις δύο) για να το αφήσουμε σε θέση που δεν συγκρούεται πριν το εκτυπώσουμε ξανά.

140 c\$(6)= "Z ZZZ ZZZZ ZZZ Z"
150 c\$(7)= "Z ZZZ ZZZZ ZZZ Z"

1533 |LOCATESP,0,yn,xn:ynn=yn:|COLAY,@cl%,0:IF cl%=0 THEN 1536
1534 yn=ya:|POKE, 27001,yn:|COLAY,@cl%,0:IF cl%=0 THEN 1536
1535 xn=xa: yn=ynn:|POKE, 27001,yn:|POKE, 27003,xn:|COLAY,@cl%,0:IF cl%=1 THEN
yn=ya:|POKE,27001,yn
1536 ya=yn:xa=xn
1537 RETURN

10.3 Πώς να ανοίξετε μια πύλη στη διάταξη

Αν θέλετε ο χαρακτήρας σας να είναι σε θέση να πάρει ένα κλειδί και να ανοίξει μια πύλη ή γενικά να αφαιρέσει ένα μέρος της διάταξης για να επιτρέψει την πρόσβαση, πρέπει να κάνετε δύο βήματα:

- 1) Έχετε ορίσει ένα sprite σκουπίσματος 8x8 όπου όλα είναι μηδενικά. Χρησιμοποιώντας |LAYOUT
το εκτυπώνετε στις θέσεις που θέλετε
- 2) Στη συνέχεια, χρησιμοποιώντας ξανά το |LAYOUT, εκτυπώνετε τα κενά στα σημεία που έχετε διαγράψει. Τότε ο χάρτης διάταξης θα μείνει με τον χαρακτήρα " " σε αυτές τις θέσεις και η συνάρτηση σύγκρουσης με τη διάταξη θα έχει ως αποτέλεσμα μηδέν.

Στο παιχνίδι "Mutant Montoya" αυτή η τεχνική χρησιμοποιείται για να ανοίξει η πύλη του κάστρου, καθώς και για να ανοίξουν οι πύλες που οδηγούν στην πριγκίπισσα.



Σχ. 62 Τροποποίηση διάταξης όταν σηκώνεται το κλειδί

Το ακόλουθο παράδειγμα απεικονίζει την έννοια ανοίγοντας μια πύλη στις συντεταγμένες (10, 12) με μέγεθος 2 μπλοκ, παίρνοντας ένα κλειδί που ορίζεται με το sprite 16.

Μόλις σηκώσετε το κλειδί, η πύλη ανοίγει και το κλειδί απενεργοποιείται για να μην αξιολογείται πλέον η σύγκρουση με αυτό, δηλαδή η εντολή |COLSP θα επιστρέψει ένα 32 από τη στιγμή που θα σηκώσετε το κλειδί, αν συγκρουστείτε ξανά με αυτό.

Μετά το άνοιγμα της πύλης, αν μετακινήσετε τον χαρακτήρα στο σημείο όπου βρισκόταν η πύλη, η σύγκρουση με τη διάταξη θα έχει ως αποτέλεσμα 0.

```
'----- αυτό το τμήμα βρίσκεται μέσα στον λογικό
Βρόχο ---- 6410 |PRINTSPALL,1,0
6411 |COLSP,@cs%,0:IF cs%<32 THEN IF cs%>=15 then gosub 6500 (
περισσότερες οδηγίες . . . . .)

'----- ρουτίνα ανοίγματος πύλης -----
6499 ' ελέγχετε ότι η σύγκρουσή σας είναι με το κλειδί, το οποίο είναι
το sprite 16 6500 delete$="MM":spaces$="      ':' το sprite διαγραφής
έχει οριστεί ως "M" (το M είναι το sprite 18 στη "γλώσσα" της εντολής
|LAYOUT)
6501 if cs%=16 then |LAYOUT,10,12,@delete$ : |LAYOUT,10,12,@spaces$:
|SETUPSP,16,0,0,0,0
6502 επιστροφή
```

10.4 Ένα παιχνίδι παζλ: LAYOUT με φόντο

Στη συνέχεια, θα δούμε ένα παράδειγμα που χρησιμοποιεί τη διάταξη και την αντικατάσταση, για το οποίο καθορίζει ένα κατώφλι ASCII που επιτρέπει στην εντολή |COLAY να μην λαμβάνει υπόψη τη σύγκρουση με τα στοιχεία φόντου. Συγκεκριμένα, το στοιχείο φόντου είναι το γράμμα "Y", το οποίο αντιστοιχεί στο sprite id= 30, και το ASCII του "Y" είναι 89.



Εικ. 63 Διάταξη με μοτίβο φόντου και αντικατάσταση

Όπως μπορείτε να δείτε στο παράδειγμα, είναι απαραίτητο να καλέσετε το COLAY με την παράμετρο threshold μόνο μία φορά, καθώς οι διαδοχικές κλήσεις λαμβάνουν ήδη υπόψη αυτό το threshold.

Μια άλλη ενδιαφέρουσα πτυχή είναι η διαχείριση του πληκτρολογίου σε αυτό το παράδειγμα. Είναι βέλτιστη για την εκτέλεση του ελάχιστου δυνατού αριθμού λειτουργιών |COLAY και ταυτόχρονα δίνει μια πολύ ευχάριστη αίσθηση όταν κινείστε σε έναν διάδρομο και συνδέεστε με έναν άλλο διάδρομο με δύο πλήκτρα πατημένα

```

10 MODE 0:DEFINT A-Z: CALL &6B78:' install RSX
21 on break gosub 5000
25 call &bc02:'restore default palette just in case'.
26 gosub 2300:' παλέτα με αντικατάσταση
30 FOR j=0 TO 31:|SETUPSP,j,0,&X0:NEXT:'επαναφορά των sprites
40 |SETLIMITS,0,80,0,200: ' όρια της οθόνης αναπαραγωγής
45 |SETUPSP,30,9,&84d0:' πλέγμα φόντου ("Y")
46 |SETUPSP,31,9,&84f2:' τούβλο ("Z")
50 dim c$(25):for i=0 to 24:c$(i)=" ":next
100 c$(1)="ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ".
110 c$(2)="ZYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYZ".
120 c$(3)="ZYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYZ".
125 c$(4)="ZYZZZYZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZYZ".
130 c$(5)="ZYZZZYZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZYYZZYZ".
140 c$(6)="ZYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYZ".
150 c$(7)="ZYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYZ".
160 c$(8)="ZYZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZYZ".
170 c$(9)="YZ".      ZYZ      ZYZ"
190 c$(10)="YZ"      ZYZ      ZYZ"

195 c$(11)="ZYZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZYZ".
200 c$(12)="ZYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYZ".
210 c$(13)="ZYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYZ".

```

```
220 c$(14)="ZYZZZYZZZZZZZZZZZZZZZZZZZ  
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZYZ".  
230 c$(15)="YYYYYYYYYYYYYYYYYYYYYYYY  
Z ZYYYYYYYYYYYYYYYYYYYYYYYYYYYYZ"  
240 c$(16)="YYYYYYYYYYYYYYYYYYYYYYYY  
Z ZYYYYYYYYYYYYYYYYYYYYYYYYYYYYZ"  
c$(17)="ZYZZZZZZZZZZZZZZZZZZYZZZ  
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZYZ".  
260 c$(18)="YYYYYYYYYYYYYYYYYYYYYYYY  
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY  
Z".  
270 c$(19)="YYYYYYYYYYYYYYYYYYYYYYYY  
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY  
Z".
```

271 c\$(20)="ZZ".
 272 c\$(21)=""
 273 c\$(22)=""
 274 c\$(23)=""

εκτυπώνουμε τη διάταξη

310 FOR i=0 TO 20:|LAYOUT,i,0,@c\$(i):NEXT
 311 locate 1,1:pen 9:print "DEMO OVERWRITE".
 312 locate 3,23:pen 11:print "BASIC using 8BP".
 320 |SETUPSP,0,0,&x01000111:' Ανίχνευση σύγκρουσης sprite και διάταξης
 330 |SETUPSP,0,7,1:dir=1: ' ακολουθία = 1 (καρύδα δεξιά)
 340 xa=20*2:xn=xa:ya=12*8:yn=ya:' συντεταγμένες του χαρακτήρα
 350 |LOCATESP,0,ya,xa: 'τοποθετήστε τον χαρακτήρα (χωρίς να τον εκτυπώσετε)
 360 |PRINTSPALL,0,1,0:' εκτυπώνει sprites
 361 cl%=0:' μεταβλητή σύγκρουσης
 362 |COLAY,89,cl%,0:'κατώτατο όριο chr\$("Y") είναι 89
 'TΟ ΠΑΙΧΝΙΔΙ ΑΡΧΙΖΕΙ
 401 |MUSIC,0,0,5
 402 ' ανάγνωση πληκτρολογίου και συγκρούσεις. αν πάμε προς την κατεύθυνση H (ή p), πρώτα ελέγξτε αν είναι πατημένο το πλήκτρο κατεύθυνσης V (q a) και αντίστροφα.
 404 if dirn <3 then gosub 450: gosub 410 else gosub 410:gosub 450
 405 |LOCATESP,0,yn,xn:|PRINTSPALL
 406 ya=yn:xa=xn
 407 Πηγαίνετε στο 404

409 ' οριζόντια κατεύθυνση πληκτρολογίου ---
 410 εάν INKEY(27)<0 τότε 430
 420 xn=xa+1poke 27003,xn:|COLAY: IF cl%=0 then if dir>>1 then |SETUPSP,0,7,1:DIR=1:xn=xa:return else dirn=1:return
 421 xn=xa:poke 27003,xn:return:'υπάρχει σύγκρουση
 430 if INKEY(34)<0 then return
 440 xn=xa-1:poke 27003,xn:|COLAY: IF cl%=0 then if dir>>2 then |SETUPSP,0,7,2:DIR=2:xn=xa:return else dirn=2:return
 441 xn=xa:poke 27003,xn:'υπάρχει σύγκρουση
 442 επιστροφή

449 κάθετη κατεύθυνση πληκτρολογίου

450 εάν INKEY(67)<0 τότε 480
 460 yn=ya-2:poke 27001,yn:|COLAY: IF cl%=0 then if dir>>3 then |SETUPSP,0,7,3:DIR=3:yn=ya:return else dirn=3:return
 461 yn=ya:poke 27001,yn:'υπάρχει σύγκρουση
 if INKEY(69)<0 then return
 490 yn=ya+2:poke 27001,yn:|COLAY: IF cl%=0 then if dir>>4 then |SETUPSP,0,7,4:DIR=4:yn=ya:return else dirn=4:return
 491 yn=ya:poke 27001,yn:'υπάρχει σύγκρουση
 492 επιστροφή

2300 REM ----- PALETA διαφανή sprites MODE 0-----
 2301 INK 0.11: REM γαλάζιο
 2302 INK 1.15: REM πορτοκαλί
 2303 INK 2,0 : INK 3,0: REM μαύρο
 2305 INK 4,26: INK 5,26: REM λευκό
 2307 INK 6,6: INK 7,6: REM κόκκινο
 2309 INK 8,18: INK 9,18: REM πράσινο
 2311 INK 10.24: INK 11.24: REM κίτρινο
 2313 INK 12,4: INK 13,4 :REM ματζέντα
 2315 INK 14,16 : INK 15, 16:πορτοκαλί REM
 2317 KITPINO=10
 2420 RETURN
 |MUSIC

5010 τέλος

10.5 Πώς να εξοικονομήσετε μνήμη στις διατάξεις σας

Αν το παιχνίδι σας έχει πολλές οθόνες και πρέπει να εξοικονομήσετε χώρο, μπορείτε να χρησιμοποιήσετε πολλές απλές τεχνικές για να εξοικονομήσετε μνήμη. Μια οθόνη καταναλώνει σχεδόν 0,5 KB, οπότε είναι σημαντικό να χρησιμοποιείτε μεθόδους για να μειώσετε το μέγεθός της

Ο ευκολότερος τρόπος για να το κάνετε αυτό είναι να επεξεργαστείτε τις οθόνες σαν να ήταν sprites. Τις επεξεργάζεστε με το SPEDIT (για παράδειγμα) και κάθε εικονοστοιχείο θα αντιπροσωπεύει ένα στοιχείο της διάταξης. Ανάλογα με το χρώμα, θα αντιπροσωπεύει βράχους, τούβλα, κενό χώρο, νερό, γη, κ.λπ. Καθώς υπάρχουν 16 διαθέσιμα χρώματα στη λειτουργία 0, θα έχετε 16 τύπους τούβλων. Το sprite που θα δημιουργήσετε θα πρέπει να το αποθηκεύσετε ως εικόνα και πριν το εμφανίσετε στην οθόνη, να το μετατρέψετε σε διάταξη, σαρώνοντας το pixel προς pixel και μετατρέποντάς το στον κώδικα ASCII που χρειάζεστε (θα πρέπει να το προγραμματίσετε σε BASIC ή ό,τι άλλο θέλετε). Αν θεωρήσουμε ότι θα χρησιμοποιήσετε 5 γραμμές χαρακτήρων για τους δείκτες του παιχνιδιού, μια οθόνη θα καταναλώνει 20x20 pixels = 400 pixels = 200 Bytes. Επομένως, σε 1KB μπορούν να χωρέσουν 5 οθόνες και σε 10KB μπορούν να χωρέσουν 50 οθόνες. Θα έχετε χρησιμοποιήσει 4 bits ανά στοιχείο διάταξης.

Η επεξεργασία οθονών σαν να ήταν sprites είναι πολύ "οπτική", αν και μπορείτε να αποφασίσετε να χρησιμοποιήσετε λιγότερα bits ανά στοιχείο διάταξης. Αν χρησιμοποιήσετε μόνο 2 bits, θα έχετε 4 τύπους στοιχείων και μπορείτε επίσης να επεξεργαστείτε οθόνες σαν να ήταν εικόνες, χρησιμοποιώντας MODE 1. Σε αυτή την περίπτωση θα μπορείτε να χωρέσετε 100 οθόνες σε 10KB. Αν χρησιμοποιείτε διαφορετικό αριθμό bits ανά τούβλο, γίνεται περίπλοκο γιατί δεν μπορείτε να σχεδιάσετε τις οθόνες σαν να ήταν sprites, αλλά μπορείτε να προγραμματίσετε κάτι που να μετατρέπει μια εικόνα που σχεδιάζετε στα bits που χρειάζεστε.

Μια άλλη απλή και αποτελεσματική λύση είναι να ορίσετε κάθε διάταξη με μεγάλα μπλοκ, π.χ. 8x16 pixels. Και να χτίζουμε τις οθόνες ορίζοντάς τες με χαρακτήρες που μπορούμε να αποθηκεύσουμε στη μνήμη. Στο βιντεοπαιχνίδι "Happy Monty", η πρώτη οθόνη του "Mutant Monty" αναδημιουργείται με μια μήτρα 16x10 χαρακτήρων = 160 bytes, έτσι ώστε 25 οθόνες να καταλαμβάνουν 4 KB. Είναι αλήθεια ότι αυτή η διάταξη έχει πλάτος μόνο 16 τετράγωνα και όχι τα 20 που μπορεί να είναι, και επίσης ορίζονται μόνο 10 τετράγωνα κάθετα, αντί για 25 που μπορεί να είναι, αλλά με αυτόν τον τρόπο καταλαμβάνει πολύ λίγη μνήμη και μπορούμε να δημιουργήσουμε πολλές οθόνες.

"IK m o JG" " IGGGGGGGH IGGGGGGGHq" "z c xo " "C CCGK d" " F oC oC IDD DDDDDDDDDD " " F C F F " " Fv C JHz b xoF " " " " F C IGGGGGGGGGGGH " " IGGH z a xw" "	
--	--

ΕΕΕΕΕοΕΕΕ "	
--------------------	--

Σχ. 64 μια διάταξη που ορίζεται με 160 χαρακτήρες

11 Προχωρημένος προγραμματισμός και "μαζική λογική".

11.1 Μέτρηση της ταχύτητας των εντολών

Ο διερμηνέας BASIC είναι πολύ βαρύς στην εκτέλεση, διότι όχι μόνο εκτελεί κάθε εντολή, αλλά αναλύει επίσης τον αριθμό γραμμής, αναλύει την εισαγόμενη εντολή, επικυρώνει την ύπαρξή της, τον αριθμό και τον τύπο των παραμέτρων, ότι οι τιμές της βρίσκονται σε έγκυρα εύρη (π.χ. το PEN 40 είναι παράνομο) και πολλά άλλα πράγματα. Είναι η συντακτική και σημασιολογική ανάλυση κάθε εντολής που πραγματικά βαραίνει και όχι τόσο η εκτέλεσή της. Η περίπτωση των εντολών RSX δεν αποτελεί εξαίρεση. Ο διερμηνέας της BASIC ελέγχει τη σύνταξή τους και αυτό βαραίνει πολύ, παρόλο που πρόκειται για ρουτίνες γραμμένες σε ASM, διότι πριν τις καλέσει, ο διερμηνέας της BASIC έχει ήδη κάνει πολλά πράγματα.

Επομένως, πρέπει να εξοικονομήσετε εκτέλεσεις εντολών με έξυπνο προγραμματισμό, ώστε η λογική του προγράμματος να περνάει από όσο το δυνατόν λιγότερες εντολές, ακόμη και αν αυτό σημαίνει μερικές φορές ότι πρέπει να γράψετε περισσότερες. Μια απαραίτητη πρακτική είναι η χρήση εντολών που μετακινούν ή επηρεάζουν μια ομάδα sprites, όπως η COLSPALL,

|AUTOALL ή |MOVERALL, αποφεύγοντας τη χρήση βρόχων με εντολές που επηρεάζουν ένα μόνο sprite.

Ένας καθοριστικός παράγοντας κατά την κλήση μιας εντολής είναι ο αριθμός των παραμέτρων. Όσο περισσότερες παραμέτρους έχει, τόσο πιο δαπανηρή είναι η ερμηνεία της για τη BASIC, ακόμη και αν πρόκειται για μια ρουτίνα ASM που καλείται με CALL, επειδή η εντολή CALL εξακολουθεί να είναι BASIC και πριν από την πρόσβαση στη ρουτίνα στην ASM, ο αριθμός και ο τύπος των παραμέτρων αναλύονται αναπόφευκτα.

Για να αξιολογήσετε το κόστος εκτέλεσης μιας εντολής μπορείτε να χρησιμοποιήσετε το ακόλουθο πρόγραμμα. Μπορείτε επίσης να το χρησιμοποιήσετε για να αξιολογήσετε την απόδοση νέων συναρτήσεων assembler που ενσωματώνετε στη βιβλιοθήκη 8BP, αν το επιθυμείτε.

```
1 κλήση &6b78
10 MNHMH 23499
11 DEFINT a-z
12 c%=0: a=2
30 FOR i=0 TO 31:|SETUPSP,i,0,0,0,0:NEXT:'reset
31 επαναλήψεις=1000
40 a!= XRONOS
50 FOR i=1 TO επαναλήψεις
60 <εδώ βάζετε μια εντολή, π.χ. PRINT "A">
70 ΕΠΟΜΕΝΟ
80 b!=XRONOS
90 PRINT (b!-a!): rem τι χρειάζεται σε μονάδες χρόνου cpc. (1/300
δευτερόλεπτα)
100 c!=((b!-a!)*1/300)/επαναλήψεις: rem c! = πόσο διαρκεί κάθε
επανάληψη σε δευτερόλεπτα
120 d!=(1/50)/c!
130 PRINT "μπορείτε να εκτελέσετε ",d!, "εντολές ανά σάρωση (1/50 sec)".
140 PRINT "η εντολή διαρκεί ";(c!*1000 -0.47); "χιλιοστά του δευτερολέπτου";
"η εντολή διαρκεί ";(c!*1000 -0.47); "χιλιοστά του δευτερολέπτου".
```

Σημείωση: Για τους ειδικούς στη γλώσσα assembly, θα πρέπει να σημειώσετε ότι αν σκοπεύετε να μετρήσετε το χρόνο εκτέλεσης μιας ρουτίνας που απενεργοποιεί εσωτερικά τις διακοπές (χρησιμοποιεί τις εντολές DI, EI), ο χρόνος που μεσολαβεί κατά την απενεργοποίηση δεν είναι μετρήσιμος με αυτό το πρόγραμμα BASIC. Οι εντολές 8BP δεν απενεργοποιούν τις διακοπές και είναι όλες μετρήσιμες.

Ας δούμε παρακάτω το αποτέλεσμα της απόδοσης ορισμένων εντολών (που μετρήθηκαν με το παραπάνω πρόγραμμα). Πρέπει να πούμε ότι είναι ταχύτερη η εκτέλεση μιας απευθείας κλήσης στη διεύθυνση μνήμης (μια CALL &XXXX) από την κλήση της αντίστοιχης εντολής RSX. Στον παρακάτω πίνακα είναι προφανές ότι όσο μικρότερο είναι το αποτέλεσμα (εκφρασμένο σε χιλιοστά του δευτερολέπτου), τόσο ταχύτερη είναι η εντολή. Ο πίνακας που παρουσιάζεται εδώ θα πρέπει να τον έχετε πάντα υπόψη σας και να παίρνετε τις προγραμματιστικές σας αποφάσεις με βάση αυτόν. Πρόκειται για έναν πίνακα με μετρήσεις των εντολών BASIC και των εντολών 8BP.

Εντολή	ms	Σχόλιο
ΕΚΤΥΠΩΣΗ "Α"	3.63	Πολύ αργή. Μην το σκέφτεστε καν να το χρησιμοποιήσετε, εκτός από περιστασιακά για να αλλάξετε τον αριθμό των ζωών, αλλά μην εκτυπώνετε σκορ σε ένα παιχνίδι για κάθε εχθρό που σκοτώνετε.
ΤΟΠΟΘΕΤΗΣΗ 1,1: Σημεία PRINT	24.8 + 7	Η τοποθέτηση του δρομέα κειμένου με LOCATE και η εκτύπωση της μεταβλητής "points" με τιμή φεγγαριού είναι πολύ δαπανηρή. Αν ενημερώνετε τα σημεία, κάντε το μόνο από καιρό σε καιρό και όχι σε κάθε κύκλο παιχνιδιού.
C\$=str\$(σημεία) PRINTAT,0, y, x, @c\$	10	Η εκτύπωση των κουκίδων με τη χρήση PRINTAT είναι πολύ πιο αποδοτική από τη χρήση LOCATE + PRINT (=32 ms) , αλλά εξακολουθεί να είναι ακριβή. Χρησιμοποιήστε το PRINTAT με φειδώ.
REM hello	0.20	Σχόλια καταναλώνουν
' hello	0.25	Εξοικονομείτε 2 bytes μνήμης, αλλά είναι πιο αργή!
GOTO 60	0.19	Πολύ γρήγορα! Ακόμη πιο γρήγορα από το REM. Χρησιμοποιήστε αυτή την εντολή ανελέητα, χρησιμοποιήστε την!!!
A = 3	0.55	Μια απλή ανάθεση κοστίζει. Όλα κοστίζουν, κάθε οδηγία πρέπει να μελετηθεί.
A = B	0.72	Η ανάθεση της τιμής μιας μεταβλητής σε μια άλλη είναι πιο δαπανηρή από την ανάθεση μιας τιμής. Και η ανάθεση της τιμής ενός πίνακα είναι ακόμη πιο ακριβή, επειδή η πρόσβαση στον πίνακα κοστίζει. Και αν ο πίνακας είναι δισδιάστατος, κοστίζει ακόμη περισσότερο.
A = miarray(x)	1.33	
A= miarray(x,y)	1.84	
 LOCATESP,i,10,20	2.8	Εάν δεν χρησιμοποιείτε αρνητικές συντεταγμένες, είναι προτιμότερο να χρησιμοποιήσετε την εντολή BASIC POKE για να ορίσετε τις συντεταγμένες.
 LOCATESP,i,y,x	3.22	Εάν οι συντεταγμένες είναι μεταβλητές, τότε χρειάζεται περισσότερος χρόνος.
CALL &XXXX,i,x,y	1.81	Το ισοδύναμο CALL είναι πολύ πιο γρήγορο.
 MOVER,31,1,1,1	3.23	Είναι κάπως αργή και πρέπει επομένως να χρησιμοποιείται με φειδώ.
ΚΛΗΣΗ &XXXX,31,1,1	1.77	Η αντίστοιχη κλήση είναι πολύ πιο γρήγορη

POKE &xxxx, τιμή	0.71	Πολύ γρήγορα! Χρησιμοποιήστε το για να ενημερώσετε τις συντεταγμένες του sprite (αν είναι θετικές). Το POKE δεν δέχεται αρνητικούς αριθμούς, αλλά μπορείτε να χρησιμοποιήσετε τον τύπο 255+x+1 αν θέλετε να εισαγάγετε έναν αρνητικό αριθμό. Για παράδειγμα, για να εισαγάγετε ένα -4 θα πληκτρολογούσατε 255-4+1=252. Ένας άλλος απλός τρόπος για να εισάγετε θετικούς και αρνητικούς αριθμούς είναι να χρησιμοποιήσετε τη διεύθυνση POKE, x και 255.
POKE dir,data	0.85	Πολύ γρήγορα αν σκεφτεί κανείς ότι πρέπει επίσης να μεταφράσει τη μεταβλητή "dir".
 POKE,&xxxx,value	2.5	Επιτρέπει αρνητικούς αριθμούς και αν ενημερώνετε μόνο μία συντεταγμένη (X ή Y) είναι καλύτερο από το LOCATESP.
X=PEEK(&xxxx)	0.93	Πολύ γρήγορα! Ανάλογα με τον τύπο του παιχνιδιού, μπορεί να αποτελέσει εναλλακτική λύση για το COLSP, εξετάζοντας το χρώμα μιας διεύθυνσης μνήμης της οθόνης. Στο παράρτημα για τη μνήμη βίντεο εξηγώ πώς να το κάνετε αυτό.
X=INKEY(27)	1.12	Πολύ γρήγορα. Κατάλληλο για βιντεοπαιχνίδια, αν και πρέπει να το χρησιμοποιείτε έξυπνα, όπως συνιστάται σε αυτό το βιβλίο.
AN x>50 TOTE x=0	1.42	Κάθε FI ζυγίζει, πρέπει να προσπαθήσουμε να τα σώσουμε γιατί μια λογική παιχνιδιού πρόκειται να έχει πολλές
IF A=value THEN GOTO 100 Vs AN A=τιμή TOTE 100	1.24 Vs 1.18	Και οι δύο προτάσεις είναι ισοδύναμες, αλλά η δεύτερη χρειάζεται λιγότερο χρόνο.
AN inkey(27)=0 τότε x=5	1.75	Αποδεκτό. Είναι πιο γρήγορο από το να κάνετε b=INKEY(27) και στη συνέχεια το IF...THEN.
10 Εάν inkey(27) τότε 30 20 x=5 30 <Οδηγίες>.	1.0	Ένας πολύ πιο αποτελεσματικός τρόπος για να κάνετε το ίδιο
AN x>0 τότε Vs AN x τότε	1.3 Vs 0.8	Στη BASIC είναι δυνατή η εξοικονόμηση 0,5ms, λαμβάνοντας υπόψη ότι κάθε μη μηδενική τιμή σημαίνει TRUE. Αν θέλουμε να ελέγξουμε μια συγκεκριμένη τιμή θα το κάνουμε: 10 IF x>20 THEN 30 20 <πράγματα που πρέπει να γίνουν αν x>20> 30 ... Η χρήση αυτής της τεχνικής συνιστάται ανεπιφύλακτα στην ανάγνωση με πληκτρολόγιο.

A=A+1: ΑΝ A>4 τότε Α=0	2.6	Είναι πολύ καλύτερο να χρησιμοποιήσετε τη δεύτερη επιλογή (χρήση MOD). Από την άλλη πλευρά, η χρήση του MOD πρέπει να γίνεται με προσοχή. Εάν το κάνουμε:
Vs	Vs	A=(A+1) MOD 3
A=A MOD 3 +1	1.7	Μας κοστίζει 2 ms, επειδή τα στηρίγματα είναι πολύ ακριβά, και όμως παίρνουμε το ίδιο πράγμα. Αν θέλουμε να μετρήσουμε από έναν αριθμό διαφορετικό από το 1, βάζουμε οποιονδήποτε περιττό αριθμό και αυτό θα το κάνει.
A=A MOD 3 + <impar>		Υπάρχει ένας ακόμα πιο γρήγορος τρόπος για να το κάνετε αυτό, με τον δυαδικό τελεστή AND, τον οποίο θα εξετάσουμε τώρα.
A=1+A AND 7 (αρχική τιμή 0 Τελική αξία 7)	1.6	Αυτό σας επιτρέπει να μεταβάλλετε μια μεταβλητή κυκλικά μεταξύ N τιμών , ώστε να μπορείτε να επιλέξετε ένα sprite ID για το νέο σας πλάνο ή για έναν εχθρό που εισέρχεται στην οθόνη.
A=20 +A MOD 7 (αρχική τιμή 0 Τελική αξία 26)	1.88	Είναι προτιμότερο να χρησιμοποιήσετε το AND παρά το MOD, επειδή το AND είναι μια γρήγορη δυαδική πράξη και το MOD περιλαμβάνει διαίρεση, η οποία είναι πολύ ακριβή για τον αγαπημένο μας μικροεπεξεργαστή Z80. Ωστόσο, αν χρειαστεί να χρησιμοποιήσουμε ID sprites που δεν αρχίζουν με 1, τότε θα χρειαστούμε παρενθέσεις, επειδή ο τελεστής "+" έχει προτεραιότητα έναντι του "AND" και το πλεονέκτημα ταχύτητας του AND χάνεται. Σε αυτή την περίπτωση
A=21 + (A και 7) (αρχική τιμή 21 Τελική αξία 28)	1.95	
		είναι καλύτερο MOD. Εν πάσῃ περιπτώσει, πάντα να το δοκιμάζετε και να επιλέγετε γιατί ανάλογα με τον αρχικό αριθμό που θα προσθέσετε θα έχετε διαφορετικούς χρόνους. Απίστευτο αλλά αληθινό 20+A mod 7 → παίρνει 1.88 29+A mod 7 → παίρνει 1.94
Εάν A<0 τότε A=15	1.71	Ελέγξτε ότι ένας αριθμός δεν είναι αρνητικός
A=A KAI 15	1.24	Μπορείτε να απλοποιήσετε τον έλεγχο επειδή ένας αρνητικός αριθμός είναι στην πραγματικότητα ένας αριθμός που έχει ένα "1" στο πιο σημαντικό bit, και αφαιρούμε την αρνητικότητα με ένα απλό και
:	0.05	Δεν εξοικονομεί πολλά, αλλά είναι ταχύτερο να χρησιμοποιείτε το ":" αντί για έναν νέο αριθμό γραμμής, και αν το εφαρμόσετε πολλές φορές, θα εξοικονομήσετε σημαντικά χρήματα. Δύο εντολές σε δύο γραμμές θα ξοδέψουν 0,03ms περισσότερο απ' ό,τι αν και τα δύο βρίσκονται στην ίδια γραμμή χωρισμένα με ":".
 PRINTSP,0,10,10,1 0	5.3	Ενιαίο sprite 14 x 24 (7 bytes x 24 γραμμές) Αν πρόκειται να εκτυπώσετε πολλά sprites, είναι πολύ καλύτερο να εκτυπώσετε όλα τα sprites ταυτόχρονα με την PRINTSPALL.

CALL &xxxx,0,10,10,10	3.5	Ισοδύναμο με το PRINTSP, επομένως ταχύτερο, αλλά λιγότερο ευανάγνωστο.
 PRINTSPALL (32 sprites 8x16 της λειτουργίας 0, δηλ. 4 bytes x 16 γραμμές)	55.4	<p>Αυτό σημαίνει περίπου 18 fps σε πλήρες φορτίο sprite. Αυτό που χρειάζεται είναι</p> $T = 3,25 + N \times 1,7$ <p>Δηλαδή, 1,7 ms ανά sprite και σταθερό κόστος 3 ms. Αυτό το σταθερό κόστος είναι το κόστος της ανάλυσης της BASIC συν το κόστος της αναζήτησης sprites για εκτύπωση στον πίνακα sprite. Αν παραλείψετε τις παραμέτρους (είναι δυνατόν και θα πάρετε τις τιμές της τελευταίας κλήσης), εξοικονομείτε 0,6 ms στο σταθερό μέρος, δηλαδή:</p> $T = 2,6 + N \times 1,1$ <p>Εάν η εκτύπωση είναι υπερτυπωμένη ή/και ανάποδα τυπωμένη, είναι ακριβότερη. Το σχετικό κόστος κάθε τύπου εκτύπωσης παρουσιάζεται παρακάτω:</p> <p>Κανονική εκτύπωση: 100%. Εκτύπωση με αντικατάσταση: 164% Εκτύπωση με αναστροφή: 179% Εκτύπωση με αναστροφή: 164% Εκτύπωση με αναστροφή: 179% Εκτύπωση με αναστροφή: 179% Εκτύπωση ανάποδα με αντικατάσταση: 220%.</p>
 PRINTSPALL,N,0,0 (χωρίς ενεργό sprite)	2.6	Κόστος παραγγελίας sprites: Όταν N=0, χωρίς sprites προς εκτύπωση, η συνάρτηση πρέπει να εκτελέσει διαδοχικά τον πίνακα sprites. Άλλα το να τον διατρέξει με τη σειρά είναι πιο ακριβό, όπως αποδεικνύεται από το χρόνο που καταναλώνεται καθώς αυξάνεται το N. Η διαφορά χρόνου (5,9 -2,6 =2,5ms) είναι αυτό που κοστίζει για να ταξινομηθούν όλα τα sprites.
 COLAY,@x%,0	3.0 Vs	Χρησιμοποιήστε το μόνο στο χαρακτήρα, όχι στους εχθρούς, αλλιώς το παιχνίδι θα επιβραδυνθεί. Αν ο χαρακτήρας είναι πολλαπλάσιο του 8, είναι πιο γρήγορος. Σε αυτό το παράδειγμα ήταν 14x24 και λογικά 14x24 ήταν το μέγεθος του χαρακτήρα.
 COLAY	2.4	δεν είναι πολλαπλάσιο του 8. Όσο μεγαλύτερο είναι το sprite τόσο περισσότερο χρόνο χρειάζεται. Αν καλέσετε την εντολή χωρίς παραμέτρους, είναι πολύ πιο γρήγορη (εξοικονομείτε 0,6 ms)!
 COLAY εναντίον ΚΑΛΕΣΤΕ &xxxx	2.4 vs 2.0	Η χρήση της CALL ως συνήθως είναι ταχύτερη, αλλά λιγότερο ευανάγνωστη.
GOSUB / RETURN	0.56	Αρκετά γρήγορα. Η μέτρηση έγινε με μια ρουτίνα που κάνει μόνο επιστροφή.

 SETUPSP, id, param, value	2.7	Αποδεκτή, αν και το POKE είναι πολύ καλύτερο για ορισμένες παραμέτρους. Υπάρχουν παράμετροι που μπορούν να οριστούν με POKE, όπως η κατάσταση, αλλά όχι άλλες (όπως μια διαδρομή). Δείτε τον οδηγό αναφοράς
FOR / NEXT	0.6	Μπορείτε να το χρησιμοποιήσετε για να περάσετε μέσα από πολλούς εχθρούς και ο καθένας από αυτούς να κινείται σύμφωνα με τον ίδιο κανόνα. Θα πρέπει να εξετάσετε αν μπορείτε να χρησιμοποιήσετε την AUTOALL ή την MOVEALL για τους σκοπούς σας, καθώς με μία εντολή θα μετακινηθούν όλοι όσοι θέλετε, κάτι που είναι πολύ καλύτερο από έναν βρόχο.
 COLSP,31, @c%	5.5	Χρειάζεται περίπου ο ίδιος χρόνος, ανεξάρτητα από τον αριθμό των ενεργών sprites. Αποφύγετε την κλήση πάντα με τη μεταβλητή σύγκρουσης για να την επιταχύνετε στα 4,3 ms.
 COLSP,31	4.3	Αν έχετε ένα σκάφος ή χαρακτήρα και πολλές βιολές, είναι πολύ πιο αποτελεσματικό να επικαλεστείτε το COLSPALL αντί να επικαλεστείτε το COLSP πολλές φορές.
 ANIMALL (αυτή η εντολή είναι διαθέσιμη μόνο με μια παράμετρο από την PRINTSPALL, δεν είναι διαθέσιμη από την μπορεί να κληθεί άμεσα)	3.5	Είναι δαπανηρή, αλλά υπάρχει τρόπος να την καλέσετε σε συνδυασμό με την κλήση PRINTSPALL , με μια παράμετρο που προκαλεί την κλήση αυτής της συνάρτησης πριν από την εκτύπωση των sprites. Αυτό εξοικονομεί το επίπεδο BASIC, δηλαδή το χρόνο που χρειάζεται για την αποστολή της εντολής, ο οποίος είναι >1ms. Έτσι μπορούμε να πούμε ότι αυτή η εντολή θα καταναλώνει κανονικά λίγο λιγότερο από 2ms.
 AUTOALL	2.76	Είναι φθηνό και μπορεί να μετακινήσει και τα 32 sprites ταυτόχρονα.
 MOVEALL,1,1	3.4	Δεν είναι πολύ ακριβό και μπορεί να μετακινήσει και τα 32 sprites ταυτόχρονα.
HXOS	10	Η εντολή ήχου "μπλοκάρει" μόλις γεμίσει το buffer των 5 νοτών. Αυτό σημαίνει ότι η BASIC λογική σας δεν πρέπει να αλυσιδωτά να δώσει περισσότερες από 5 εντολές SOUND αλλιώς θα σταματήσει μέχρι να τελειώσει κάποια νότα... Αν αποφασίσετε να τη χρησιμοποιήσετε, πρέπει να είστε πολύ προσεκτικοί γιατί είναι πολύ καταναλωτική. χρόνος εκτέλεσης (10 ms είναι πολύ μεγάλος)
AN a>1 KAI a>2 TOTE a=2 Versus EAN a>1 TOTE EAN a>2 TOTE a=2	2.52 Vs 2.39	Ένας απλός τρόπος εξοικονόμησης 0,13 ms Σε ό,τι προγραμματίζετε, να έχετε κατά νου αυτές τις λεπτομέρειες, κάθε εξοικονόμηση είναι σημαντική.

A=RND*10	4.2	Η λειτουργία BASIC RND είναι πολύ ακριβή. Μπορείτε να τη χρησιμοποιήσετε, αλλά όχι σε κάθε κύκλο παιχνιδιού, αλλά μόνο κάποια στιγμή, για παράδειγμα, όταν μια νέα εχθρός ή κάτι παρόμοιο. Μια άλλη απλή λύση είναι να αποθηκεύσετε
		10 τυχαίους αριθμούς σε έναν πίνακα και να τους χρησιμοποιήσετε αντί να καλέσετε το RND
Όριο <x>	0.75	Αρκετά γρήγορα. Χρήσιμο για χρήση σε συνδυασμό με κάποιο είδος σύγκρουσης sprite, ενισχύοντας το εκρηκτικό αποτέλεσμα.
IF a AND 7 then 30	1.19	Έχω βάλει τον χρόνο εκτέλεσης όταν πληρούται η συνθήκη. Και οι δύο περιπτώσεις είναι αρκετά γρήγορες.
IF A MOD 8 τότε 30	1.29	
ON x GOTO L1,L2,L3,L4	3.67	Μια εντολή ON GOTO είναι κατά μέσο όρο 1 ms ταχύτερη από την αντίστοιχη με τις εντολές "IF", αν και εξαρτάται επίσης από την πιθανότητα εμφάνισης κάθε μίας από τις 4 τιμές. Εάν η πιθανότητα εμφάνισης των 4 τιμών είναι η ίδια, μπορούμε να εξοικονομήσουμε 1 ms με τη χρήση της εντολής ON GOTO
Vs 60 αν x >2 τότε 63 61 if x=1 then 70 62 goto 70 63 αν x=3 τότε 70 64 επιλέξτε 70	Vs 4.8	Μπορεί να βελτιστοποιηθεί περαιτέρω ως εξής 100N X GOTO 30,40,50 20 <Οδηγίες περίπτωση x=4>: GOTO 60 30 <εντολές περίπτωση x=1>: GOTO 60 40 <Οδηγίες περίπτωση x=2>: GOTO 60 50 <Οδηγίες περίπτωση x=3>: GOTO 60 60 συνέχιση του προγράμματος Με αυτή τη στρατηγική έχουμε πέσει στα 3,54 ms.

Πίνακας 5 Κατάλογος χρόνων εκτέλεσης ορισμένων εντολών

Σημαντικές συστάσεις:

- Χρησιμοποιήστε DEFINT A-Z στην αρχή του προγράμματος.** Η απόδοση θα βελτιωθεί σημαντικά. Αυτό είναι σχεδόν υποχρεωτικό. Αυτή η εντολή διαγράφει όλες τις μεταβλητές που υπήρχαν πριν και αναγκάζει όλες τις νέες μεταβλητές να είναι ακέραιες, εκτός αν υποδεικνύεται διαφορετικά από τροποποιητές όπως "\$" ή "!" (Δείτε τον οδηγό αναφοράς του προγραμματιστή Amstrad BASIC). Σημειώστε ότι όταν χρησιμοποιείτε το DEFINT, αν θέλετε να συσχετίσετε έναν αριθμό μεγαλύτερο από 32768 θα πρέπει να το κάνετε σε δεκαεξαδικό σύστημα.
- Αν μπορείτε να αποφύγετε να περάσετε από ένα IF εισάγοντας ένα GOTO, θα είναι πάντα προτιμότερο από το
- Όταν σας λείπει η ταχύτητα και χρειάζεστε λίγη περισσότερη ταχύτητα,

χρησιμοποιήστε το **CALL**.

<διεύθυνση> αντί για RSX. Αν το κάνετε αυτό, πρέπει να περάσετε παραμέτρους που περιέχουν αρνητικούς αριθμούς σε δεκαεξαδική μορφή.

- Μην συγχρονίζετε την εντολή **PRINTSPALL** με τη σάρωση οθόνης, εκτός αν το παιχνίδι σας εκτελείται πολύ γρήγορα. Ο συγχρονισμός μπορεί να μειώσει τα FPS σας. Σε γενικές γραμμές, εφόσον έχετε 12 FPS, το παιχνίδι σας θα είναι "playable".

- Εξαλείψτε το κενό διάστημα.** Κάθε κενό διάστημα στην καταχώριση BASIC καταναλώνει 0,01ms κατά την εκτέλεση.
- Συντομεύστε το όνομα των μεταβλητών σας.** Όσο πιο μεγάλες είναι, τόσο περισσότερο κοστίζουν η πρόσβαση σε αυτές.

λειτουργία	Καιρός
A=A+1	Ένα γράμμα, διαρκεί 1,18ms
HO=HO+1	Δύο γράμματα, διαρκεί 1,2ms (2% περισσότερο)
HELLO=HELLOA+1	5 γράμματα, διαρκεί 1,25 ms (6% περισσότερο)
HELLOFRIENDS=HELLOFRIENDS +1	10 γράμματα 1,34 ms (13% περισσότερο)

- Μειώστε τον αριθμό των μεταβλητών.** Εάν υπάρχουν πολλές μεταβλητές, οι προσπελάσεις ανάγνωσης και εγγραφής είναι πιο αργές.
- Αφού καλέσετε με παραμέτρους την εντολή **|STARS** ή την εντολή **|PRINTSPALL**, ή **|COLAY** ή άλλες εντολές 8BP, τις επόμενες φορές δεν την καλείτε με παραμέτρους. Η βιβλιοθήκη 8BP έχει "μνήμη" και θα χρησιμοποιήσει τις τελευταίες παραμέτρους που χρησιμοποιήσατε. Αυτό εξοικονομεί χιλιοστά του δευτερολέπτου περνώντας από το επίπεδο ανάλυσης του διερμηνέα BASIC.
- Να έχετε πάντα κατά νου ότι μια μη μηδενική έκφραση είναι ΑΛΗΘΗΣ. Αυτό θα σας επιτρέψει να εξοικονομήσετε 0,5ms σε κάθε IF και μπορεί να χρησιμοποιηθεί στην ανάγνωση του πληκτρολογίου και στον έλεγχο μεταβλητών.

Κακή επιλογή	Καλή επιλογή (εξοικονομεί 0,5ms)
AN x<>0 TOTE <οδηγίες>	AN x TOTE <οδηγίες>
AN x=20 TOTE...	10 Εάν x=20 TOTE 30 20 <Οδηγίες>. 30
AN INKEY(34)=0 TOTE <οδηγίες>.	10 IF INKEY(34) THEN 30 20 <Οδηγίες>. 30

- Σε παιχνίδια πλοίων όπου δεν χρησιμοποιείτε το overwrite, βεβαιωθείτε ότι το πλοίο σας είναι sprite 31, έτσι ώστε να πάει "πάνω" από τα sprites που προσποιούνται ότι είναι το φόντο, καθώς το πλοίο σας θα εκτυπωθεί στη συνέχεια.
- Δοκιμή εναλλακτικών εκδοχών της ίδιας λειτουργίας
A=A+1:AN A>4 τότε A=0 : REM αυτό καταναλώνει 2,6 ms
ms A=A MOD 3 +1 : REM αυτό καταναλώνει 1,84 ms
A=1 + A AND 3 : REM αυτό καταναλώνει 1,6 ms
- Αποφύγετε τη χρήση αρνητικών συντεταγμένων. Αυτό θα σας επιτρέψει να χρησιμοποιήσετε το POKE για να ενημερώσετε τη θέση του χαρακτήρα σας. Η εντολή POKE (η εντολή της BASIC) είναι πολύ γρήγορη αλλά υποστηρίζει μόνο θετικούς αριθμούς, όπως και η PEEK. Σε περίπτωση που χρησιμοποιείτε

εγγενείς συντεταγμένες, χρησιμοποιήστε τις εντολές **|POKE** και **|PEEK** (εντολές 8BP). Διατηρήστε τη χρήση της **|LOCATESP** για την περίπτωση που πρόκειται να τροποποιήσετε και τις δύο συντεταγμένες ταυτόχρονα και μπορούν να είναι θετικές ή/και αρνητικές. Να θυμάστε επίσης ότι ένα POKE μιας αρνητικής τιμής x μπορεί να γίνει με τη χρήση της διεύθυνσης **POKE**, 255+x+1. Σε περίπτωση που θέλετε να χρησιμοποιήσετε αρνητικές συντεταγμένες για να δείξετε πώς οι εχθροί εισέρχονται αργά στην οθόνη από την αριστερή πλευρά (clipping), μπορείτε να αποφύγετε τις συντεταγμένες

Η χρήση ενός **|SETLIMITS** και έτσι παράγεται το ίδιο αποτέλεσμα με συντεταγμένες που ξεκινούν από το μηδέν και μια ελαφρώς μικρότερη οθόνη παιχνιδιού.

- Αν πρέπει να ελέγξετε κάτι, μην το ελέγχετε σε κάθε κύκλο παιχνιδιού. Μπορεί να είναι αρκετό να ελέγχετε αυτό το "κάτι" κάθε 2 ή 3 κύκλους, χωρίς να χρειάζεται να το ελέγχετε κάθε κύκλο. Για να μπορείτε να επιλέξετε πότε θα εκτελέσετε κάτι, χρησιμοποιήστε την "σπονδυλωτή αριθμητική". Στη BASIC έχετε την εντολή MOD, η οποία είναι ένα εξαιρετικό εργαλείο. Για παράδειγμα, για να εκτελέσετε μία από τις 5 φορές μπορείτε να κάνετε: **IF cycle MOD 5 = 0 THEN ...** αν και είναι προτιμότερο να χρησιμοποιείτε πράξεις **AND** παρά πράξεις **MOD**.
- Χρησιμοποιήστε τις "**ακολουθίες θανάτου**". Αυτό θα σας επιτρέψει να αποθηκεύσετε οδηγίες για να ελέγξετε αν ένα sprite που εκρήγνυται έχει φτάσει στο τελευταίο του καρέ κινούμενης εικόνας για να το απενεργοποιήσετε.
- Η επανεγγραφή είναι ακριβή: αν μπορείτε να φτιάξετε το παιχνίδι σας χωρίς επανεγγραφή, θα εξοικονομήσετε χλιοστά του δευτερολέπτου και θα κερδίσετε χρώμα. Χρησιμοποιήστε το όταν το χρειάζεστε, αλλά όχι χωρίς λόγο.
- Οι μακροεντολές κίνησης σας εξοικονομούν γραμμές BASIC επειδή δεν χρειάζεται να ελέγχετε την κατεύθυνση κίνησης του sprite. Χρησιμοποιήστε τις όποτε μπορείτε.

11.2 Το Amstrad μας έχει μόνο 64KB και αν αφαιρέσετε τη μνήμη βίντεο, τη μνήμη για τα sprites, τη μουσική και τη βιβλιοθήκη 8BP, σας μένουν 24KB BASIC για χρήση. πολύ καλά. Αν κάθε οθόνη έχει το δικό της "πρόγραμμα" στο παιχνίδι σας, μετά βίας θα είστε σε θέση να

να φτιάξετε ένα σετ 10 οθονών.

Υπάρχουν δύο πράγματα που πρέπει να προσπαθήσετε να κάνετε για να μειώσετε τη χρήση μνήμης

- Δημιουργία οθονών χαμηλού byte
- Δημιουργήστε μια ενιαία λογική που θα διέπει όλες τις οθόνες, δηλαδή ο ίδιος κύκλος παιχνιδιού θα εκτελείται σε όλες τις οθόνες του παιχνιδιού.

Στα παιχνίδια με πέρασμα οθόνης που χρησιμοποιούν διάταξη, κάθε οθόνη μπορεί να καταλαμβάνει 20x25 bytes, δηλαδή 500 bytes. Αν χρησιμοποιήσετε ορισμένα "κόλπα", όπως εξηγείται στο κεφάλαιο 8, μπορείτε να μειώσετε αυτή τη μνήμη. Στο βιντεοπαιχνίδι "Happy Monty", κατασκευάζονται 25 οθόνες με μόνο 160 bytes η καθεμία και υπάρχει μια ενιαία λογική κύκλου παιχνιδιού για όλες τις οθόνες.

Είναι πολύ σημαντικό να προγραμματίσετε μια ενιαία λογική κύκλου παιχνιδιού και να την εφαρμόσετε σε όλες τις οθόνες. Αν προγραμματίσετε μια λογική κύκλου παιχνιδιού για κάθε οθόνη, ο πηγαίος κώδικας του προγράμματός σας θα είναι τεράστιος και θα μπορείτε να προγραμματίσετε λίγες οθόνες, επειδή σύντομα θα ξεμείνετε από μνήμη.

Μπορείτε επίσης να ζεπεράσετε τους περιορισμούς μνήμης χρησιμοποιώντας αλγόριθμους που δημιουργούν λαβύρινθους ή οθόνες χωρίς να χρειάζεται να τις

αποθηκεύσετε. Με αυτόν τον τρόπο μπορείτε να φτιάξετε πολύ περισσότερες οιθόνες. Αυτό απαιτεί δημιουργικότητα, φυσικά, αλλά είναι εφικτό. Ένας αλγόριθμος

καταλαμβάνει πάντα λιγότερο χώρο από τα δεδομένα που παράγει, αν και λογικά χρειάζεται περισσότερο χρόνο για να εκτελεστεί από την απλή ανάγνωση αποθηκευμένων δεδομένων.

Μπορείτε να επαναχρησιμοποιήσετε τη λογική του εχθρού από τη μία οθόνη στην άλλη, εξοικονομώντας γραμμές κώδικα. Επωφεληθείτε από το μηχανισμό GOSUB/RETURN για το σκοπό αυτό. Είναι επίσης πολύ χρήσιμο να χρησιμοποιείτε διαδρομές στους εχθρούς. **Με το μηχανισμό διαδρομών, ο εχθρός κινείται χωρίς να εκτελεί λογική BASIC** και θα λειτουργεί πολύ γρήγορα. Απλά αναθέστε μια διαδρομή σε έναν εχθρό και αυτός θα την εκτελεί ξανά και ξανά χωρίς την ανάγκη για ακριβές δηλώσεις IF, αναθέσεις κ.λπ.

Μπορείτε επίσης να φτιάξετε παιχνίδια που φορτώνουν σε στάδια, ώστε να μην έχετε ολόκληρο το παιχνίδι στη μνήμη σας ταυτόχρονα. Αυτό είναι λίγο ενοχλητικό για τον χρήστη ταινίας (CPC464), αλλά όχι για τον χρήστη δίσκου (CPC6128).

Χρησιμοποιήστε το **sprite flipping** για να εξοικονομήσετε μνήμη στα sprites σας

11.3 Τεχνική "Mass Logic"

Συχνά θα χρειαστεί να μετακινήσετε πολλά sprites, ειδικά σε παιχνίδια τύπου space arcade ή "commando" (το κλασικό παιχνίδι της Capcom του 1985).

Θα μπορούσατε να ενεργείτε ξεχωριστά στις συντεταγμένες όλων των sprites και να τις ενημερώνετε χρησιμοποιώντας POKE, αλλά θα ήταν πολύ αργό και ανέφικτο αν θέλετε ρευστότητα στην κίνηση. Ο καλύτερος (και ευκολότερος) τρόπος είναι να κάνετε συνδυασμένη χρήση των λειτουργιών αυτόματης κίνησης και σχετικής κίνησης, οι οποίες είναι |AUTOALL και |MOVEALL αντίστοιχα.

Το κλειδί για την επίτευξη ταχύτητας σε πολλά sprites είναι η χρήση της τεχνικής που έχω ονομάσει "μαζική λογική". Αυτή η τεχνική αφορά ουσιαστικά την εκτέλεση λιγότερης λογικής ανά κύκλο παιχνιδιού (που ονομάζεται "μείωση της υπολογιστικής πολυπλοκότητας") και υπάρχουν διάφορες επιλογές για να το κάνετε αυτό:

- Χρησιμοποιήστε **μια ενιαία λογική** που επηρεάζει πολλά sprites ταυτόχρονα (χρησιμοποιώντας τις σημαίες αυτόματης ή/και σχετικής κίνησης).
- Εκτελέστε πολλές εργασίες, αλλά μόνο μία ή λίγες από αυτές σε κάθε κύκλο παιχνιδιού, χρησιμοποιώντας **αρθρωτή αριθμητική** (ή δυαδικές πράξεις) **σε κλιμακωτή διάταξη**.
- Εισάγετε **περιορισμούς στο παιχνίδι που δεν είναι σημαντικοί** ή δεν επηρεάζουν το παιχνίδι, για να μειώσετε τον αριθμό των εργασιών που εκτελούνται σε κάθε κύκλο παιχνιδιού ή για να απλοποιήσετε τις εργασίες ώστε να εκτελούνται ταχύτερα.
- Κατά γενικό κανόνα, μειώστε τον αριθμό των εντολών που εκτελεί το πρόγραμμά σας σε κάθε κύκλο παιχνιδιού, αντικαθιστώντας μερικές φορές αλγορίθμους με προ-υπολογισμούς ή βάζοντας περισσότερες εντολές ώστε (παραδόξως) να εκτελούνται λιγότερες σε κάθε κύκλο.

Αυτές οι ιδέες έχουν τον ίδιο στόχο: να εκτελείται λιγότερη λογική σε κάθε κύκλο, επιτρέποντας σε όλα τα sprites να κινούνται ταυτόχρονα, αλλά λαμβάνοντας λιγότερες αποφάσεις σε κάθε κύκλο του παιχνιδιού. Αυτό ονομάζεται **"μείωση της**

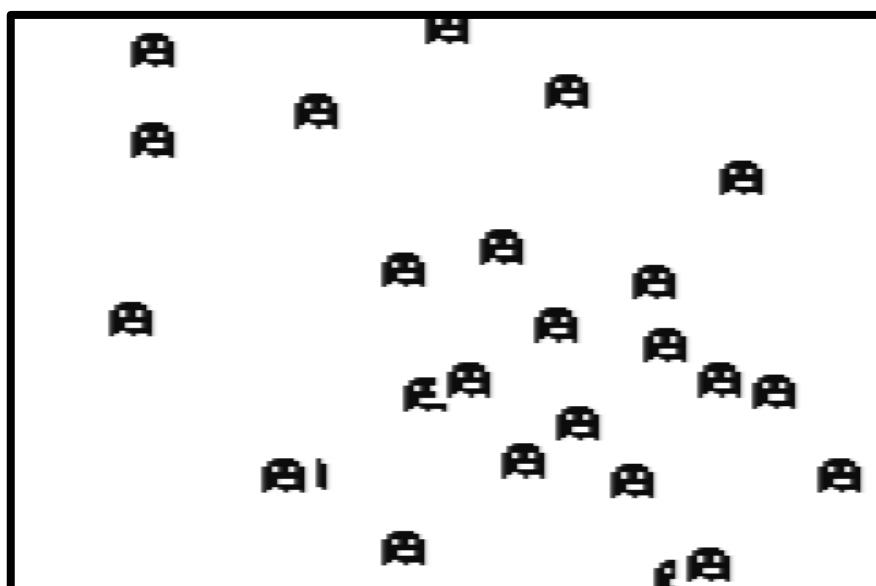
υπολογιστικής πολυπλοκότητας", μετατρέποντας ένα πρόβλημα τάξης N (N sprites) σε πρόβλημα τάξης 1 (μία μόνο λογική που πρέπει να εκτελεστεί σε κάθε πλαίσιο).

Το κλειδί είναι να καθορίσετε ποια λογική ή λογικές θα εκτελούνται σε κάθε κύκλο. Στην απλούστερη περίπτωση, αν έχουμε N sprites θα τρέξουμε απλά μία από τις N λογικές. Αλλά σε πιο σύνθετες περιπτώσεις θα πρέπει να είμαστε έξυπνοι στον καθορισμό των λογικών που θα τρέξουν.

11.3.1 Μετακινεί 32 sprites με μαζική λογική

Ας δούμε τώρα ένα απλό παράδειγμα μετακίνησης 32 sprites ταυτόχρονα και ομαλά (στα 14fps). Είναι απολύτως εφικτό. Μόνο ένα φάντασμα θα λαμβάνει αποφάσεις σε κάθε κύκλο, αν και όλα τα φαντάσματα θα κινούνται σε όλους τους κύκλους. Μπορούμε επίσης να τα κινήσουμε όλα (συσχετίζοντάς τα με μια ακολουθία κίνησης και χρησιμοποιώντας

`|PRINTSPALL,1,0`) και θα εξακολουθεί να είναι ομαλό, αλλά θα εξακολουθεί να φαίνεται ότι υπάρχει περισσότερη κίνηση, καθώς το χτύπημα των φτερών μιας μύγας (για παράδειγμα) δημιουργεί μεγάλη αίσθηση κίνησης.



Εικ. 65 με massive logics μπορείτε να μετακινήσετε 32 sprites ταυτόχρονα

Αυτό που κάναμε είναι να μειώσουμε την υπολογιστική πολυπλοκότητα. Ξεκινήσαμε με ένα πρόβλημα "τάξης N", όπου N είναι ο αριθμός των sprites. Υποθέτοντας ότι η λογική κάθε sprite απαιτεί 3 εντολές BASIC, κατ' αρχήν θα έπρεπε να εκτελεστούν $N \times 3$ εντολές σε κάθε κύκλο. Με την τεχνική "bulk logic", μετατρέπουμε το πρόβλημα "τάξης N" σε πρόβλημα "τάξης 1". Ένα πρόβλημα "τάξης 1" είναι ένα πρόβλημα που περιλαμβάνει σταθερό αριθμό πράξεων ανεξάρτητα από το μέγεθος του προβλήματος. Σε αυτή την περίπτωση έχουμε μεταβεί από $N \times 3 = 32 \times 3 = 96$ πράξεις BASIC σε μόνο 3 πράξεις BASIC. Αυτή η μείωση της πολυπλοκότητας είναι το κλειδί για την υψηλή απόδοση της τεχνικής της μαζικής λογικής.

1 MODE 0

10 MNHMH 24999: ΚΛΗΣΗ &6B78

20 DEFINT a-z

25 ' επαναφορά εχθρών

30 FOR j=0 TO 31:|SETUPSP,j,0,&X0:NEXT

35 ' num εχθροί 12 x 16 (πλάτος 6bytes x 16 γραμμές)

36 num=32: x%=0:y%=0

40 FOR i=0 TO num-1:|SETUPSP,i,9,&8ee2: |SETUPSP,i,0,&X1111:

41 |LOCATESP,i,rnd*200,rnd*80

42 επόμενο

```

43 i=0
45 gosub 100
46 i=i+1: αν i=num τότε
i=0
50 |PRINTSPALL,0,0
60 |AUTOALL
70 goto 45
100 |peek,27001+i*16,@y%
110 |peek,27003+i*16,@x%
120 if y%<=0 then |SETUPSP,i,5,2:|SETUPSP,i,6,0: return
130 if y%>=190 then |SETUPSP,i,5,-2:|SETUPSP,i,6,0: return
140 if x%<=0 then |SETUPSP,i,5,0:|SETUPSP,i,6,1: return
150 if x%>=76 then |SETUPSP,i,5,0:|SETUPSP,i,6,-1: return

160 πιθανότητα=rnd*3
170 if random=0 then |SETUPSP,i,5,2:
|SETUPSP,i,6,0:return
180 if random=1 then |SETUPSP,i,5,-
2:|SETUPSP,i,6,0:return
190 if random=2 then
|SETUPSP,i,5,0:|SETUPSP,i,6,1:return
200 if random=3 then |SETUPSP,i,5,0:|SETUPSP,i,6,1: return

```

~~4.3.2 Εναλλασσόμενη και τρεις διαδικασίες για την καταρράκτη~~

1: return.

Δεν χρειάζεται να εκτελείτε όλες τις εργασίες σε κάθε κύκλο παιχνιδιού. Για παράδειγμα, αν θέλετε να ελέγξετε αν η βολή έχει φύγει από την οθόνη, μπορείτε να ελέγχετε κάθε δύο ή τρεις κύκλους, αντί να ελέγχετε κάθε κύκλο.

Μπορείτε επίσης να κάνετε τους εχθρούς να πυροβολούν κάθε λίγους κύκλους και όχι κάθε κύκλο (διαφορετικά θα σας πυροβολούσαν πολύ!!).

Εν ολίγοις, υπάρχουν πράγματα που δεν χρειάζεται να κάνετε σε κάθε κύκλο και μπορείτε να εξοικονομήσετε την εκτέλεση εντολών ανά κύκλο και επομένως να επιτύχετε υψηλότερη ταχύτητα. Αυτό είναι στην πραγματικότητα το βασικό θεμέλιο της τεχνικής της "μαζικής λογικής".

Υπάρχουν δύο βασικές τεχνικές γι' αυτό: Η χρήση της αρθρωτής αριθμητικής και των δυαδικών πράξεων **AND**. Οι δυαδικές πράξεις **AND** είναι ταχύτερες από τις πράξεις **MOD**.

Τεχνική	Χρόνος που καταναλώνεται
A = A+1: εάν A =5 τότε A=0: GOSUB <routine>.	2,6 ms
ΑΝ κύκλος MOD 5 =0 ΤΟΤΕ gosub ρουτίνα	1,84ms, υποθέτοντας ότι έχετε ήδη μια μεταβλητή που ονομάζεται cycle η οποία ενημερώνεται

Η λειτουργία **MOD** είναι κάπως δαπανηρή και επομένως μια δυαδική λειτουργία είναι μερικές φορές καλύτερη.

Υποθέτοντας ότι έχετε τη μεταβλητή κύκλου που ενημερώνεται κάθε φορά, μπορούμε να κάνουμε μια δυαδική πράξη για να δούμε πότε μια ομάδα bits δίνει μια συγκεκριμένη τιμή. Για παράδειγμα, αν εξετάσουμε τα 4 λιγότερο σημαντικά bits της μεταβλητής κύκλου, θα πηγαίνουν πάντα από το 0000 στο 1111 και πάλι πίσω. Λοιπόν,

αν κάνουμε ένα AND 15 με αυτή τη μεταβλητή μπορούμε να κάνουμε το ίδιο όπως και με το MOD 15. Ο αριθμός 15 στο δυαδικό είναι 1111 και έτσι ένα AND αποκαλύπτει την τιμή αυτών των 4 bits.

Τεχνική	Χρόνος που καταναλώνεται
Εάν κύκλος AND 15=0 τότεgosub ρουτίνα	1,6 ms (εκτελείται μία στις 16 φορές)
Εάν κύκλος AND 1=0 τότεgosub ρουτίνα	1,6ms (Εκτελείται μία φορά κάθε 2 φορές)

Αν έχετε πολλά περιοδικά πράγματα να εκτελέσετε, μπορείτε να το κάνετε ως εξής:

c=κύκλος AND 15 : rem To 15 είναι στο δυαδικό 1111
IF c=0 THEN GOSUB <routine1> (η ρουτίνα1 εκτελείται μία φορά κάθε 16 φορές) iF c=8 THEN GOSUB <routine2>... (η ρουτίνα2 εκτελείται μία φορά κάθε 16 φορές, αλλά σε μεγάλη χρονική απόσταση από την εκτέλεση της ρουτίνας1)

Με αυτόν τον τρόπο κατανέμετε το χρόνο σε διάφορες εργασίες, έτσι ώστε σε κάθε κύκλο να κάνετε μόνο μία εργασία, αλλά μετά από αρκετούς κύκλους να έχετε κάνει όλες τις εργασίες.

Για να ελέγξετε τη μεταβλητή κύκλου και να αποφασίσετε την εκτέλεση μιας εργασίας, υπάρχει ένας καλύτερος τρόπος εκτέλεσης των δυαδικών πράξεων και είναι ο εξής:

Τεχνική	Χρόνος που καταναλώνεται
10 Αν κύκλος και 7 τότε 30 20 <Οδηγίες η οποία είναι εκτελούνται κάθε 8 κύκλους> 30 <συνέχιση προγράμματος>	1,18 ms. Αυτή είναι αναμφίβολα η καλύτερη στρατηγική για την περιοδική εκτέλεση εργασιών.

Η εφαρμογή της ίδιας στρατηγικής στο MOD αυξάνει επίσης την ταχύτητα, αν και λιγότερο από ό,τι με το AND. Ωστόσο, είναι πολύ καλή γιατί λειτουργεί, ακόμη και αν η περίοδος δεν είναι πολλαπλάσιο του 2 (μπορείτε να βάλετε MOD 7, MOD 5, MOD 10, κ.λπ.).

Τεχνική	Χρόνος που καταναλώνεται
10 Εάν ο κύκλος MOD 8 τότε 30 20 <Οδηγίες η οποία είναι εκτελούνται κάθε 8 κύκλους> 30 <συνέχιση προγράμματος>	1,29 ms. Σχεδόν τόσο καλή όσο το AND και η καλύτερη στρατηγική όταν χρειαζόμαστε μια περίοδο που δεν είναι πολλαπλάσιο του 2.

Όπως μπορείτε να δείτε, για κάθε εργασία που θέλουμε να εισάγουμε στη λογική, πρέπει να εισάγουμε ένα "IF". **Ωστόσο, μπορεί ακόμα να βελτιωθεί** και να γίνει πιο αποδοτική, χρησιμοποιώντας χρονικά διαστήματα εκτέλεσης εργασιών που είναι πολλαπλάσια. Εάν είναι πολλαπλάσια, το "IF" της εργασίας 2 μπορεί να γίνει μέσα στην εργασία 1, και το "IF" της εργασίας 3 μέσα στην εργασία 2, σε "καταρράκτη". Αυτό μειώνει σημαντικά τον αριθμό των IF που εκτελούμε σε κάθε κύκλο, καθώς σε πολλές περιπτώσεις είναι ένα μόνο IF.

Ας δούμε ένα πλήρες παράδειγμα, το οποίο σας επιτρέπει να εκτελείτε 4 διαφορετικές εργασίες, μειώνοντας όμως τον αριθμό των εργασιών που εκτελούνται ταυτόχρονα. Η παρακάτω ακολουθία αναπαριστά τη σειρά εκτέλεσης των εργασιών και στη συνέχεια τον πηγαίο κώδικα.



```
10 IF cycle AND 1 THEN 90
20 REM κάθε δύο κύκλους μπαίνουμε εδώ
25 IF cycle AND 3 THEN 80
30 REM κάθε 4 κύκλους μπαίνουμε εδώ
35 IF cycle AND 7 THEN 70
40 REM κάθε 8 κύκλους μπαίνουμε εδώ
50 <εργασία 4> : GOTO 100
70 <εργασία 3> : GOTO 100
```

80 <εργασία 2> : GOTO 100

90 <εργασία 1>.

100 REM --- τέλος εργασιών ---

Σε αυτό το παράδειγμα έχουμε επιλέξει τα διαστήματα 2, 4 και 8.

KAI 1 : αυτό μου δίνει ένα διάστημα 2 επειδή είναι μηδέν κάθε 2

κύκλους KAI 3 : είναι μηδέν κάθε 4 κύκλους

AND 7 : είναι μηδέν κάθε 8 κύκλους

Επειδή έχουμε επιλέξει λειτουργίες σε διαστήματα multiplo, οι IFs εκτελούνται "σε καταγισμό": εισερχόμαστε σε ένα IF μόνο αν έχουμε εισέλθει στο προηγούμενο:

- Οι μισοί από τους κύκλους εκτελούν ένα μόνο IF (γραμμή 10).
- Οι μισοί από τους κύκλους εκτελούν δύο εντολές IF (γραμμές 10 και 25), εκ των οποίων οι μισοί (δηλαδή το 25%) θα εκτελέσουν τρεις εντολές IF (γραμμές 10, 25 και 35).

Κατά μέσο όρο, εκτελούνται $1*50\%+2*25\%+3*25\% = 1,75$ δηλώσεις IF ανά κύκλο.

Χάρη σε αυτή τη στρατηγική της χρήσης αρθρωτής αριθμητικής με δυαδικές πράξεις σε πολλαπλά διαστήματα για την κλιμάκωσή τους, μπορούμε να μειώσουμε τον αριθμό των πράξεων "IF" στο ελάχιστο και ταυτόχρονα να μειώσουμε την υπολογιστική πολυπλοκότητα από τάξη N (n εργασίες) σε τάξη 1 (μία εργασία ανά κύκλο). Αυτό επιταχύνει πάρα πολύ τα παιχνίδια σας.

11.3.3 Απλό παράδειγμα μαζικής λογικής

Στο βιντεοπαιχνίδι "Mutante Montoya", τα sprites των εχθρών εναλλάσσονται για να τρέξουν στους διάφορους κύκλους του παιχνιδιού. Όταν προγραμμάτισα αυτό το παιχνίδι, δεν είχα ακόμα προγραμματίσει τον μηχανισμό |ROUTEALL που επιτρέπει την ανάθεση σταθερών τροχιών στα sprites, αλλά κατάφερα να το λύσω με μαζική λογική. Σε περίπτωση που θέλατε να φτιάξετε ένα παιχνίδι όπου οι εχθροί είχαν "νοημοσύνη", μια σταθερή διαδρομή δεν θα λειτουργούσε, οπότε ακόμα και αν έχετε την εντολή ROUTEALL, θα έπρεπε να περιστρέψετε τη λογική των sprites όπως περιγράφεται παρακάτω, οπότε αυτό το παράδειγμα είναι ενδιαφέρον.

Ας υποθέσουμε ότι έχουμε 3 εχθρικούς στρατιώτες που κινούνται από τα δεξιά προς τα αριστερά και από τα αριστερά προς τα δεξιά. Για να κερδίσουμε ταχύτητα θα εκτελούμε μόνο τη λογική ενός στρατιώτη σε κάθε κύκλο παιχνιδιού.

Για να διατηρήσουμε τη συντεταγμένη x κάθε στρατιώτη να κινείται προς τα εμπρός παρά το γεγονός αυτό, θα χρησιμοποιήσουμε τη σημαία αυτόματης κίνησης αντί να την ενημερώνουμε εμείς οι ίδιοι.

10 MNHMH 24999

20 MODE O: DEFINT A-Z: CALL &6B78:' install RSX

25 FOR j=0 TO 31:|SETUPSP,j,0,&X0:NEXT:'επαναφορά των sprites

26 |SETLIMITS,0,80,0,0,0,200

30 "παραμετροποίηση 3 στρατιωτών

40 dim x(3):x%=0

50 x(1)=10:xmin(1)=10:xmax(1)=60:

y(1)=60:direccion(1)=0:|SETUPSP,1,7,9 :|SETUPSP,1,0,&x1111:

|SETUPSP,1,5,0: |SETUPSP,1,6,1

60 x(2)=20:xmin(1)=15:xmax(2)=40:

y(2)=100:direccion(2)=1:|SETUPSP,2,7,10 :|SETUPSP,2,0,&x1111:

|SETUPSP,2,5,0: |SETUPSP,2,6,-1

```

70 x(3)=30:xmin(1)=5:xmax(3)=50:
y(3)=130:direccion(3)=0:|SETUPSP,3,7,9 :|SETUPSP,3,0,&x1111:
|SETUPSP,3,5,0: |SETUPSP,3,6,1
80 for i=1 to 3:|LOCATESP,i,y(i),x(i):next: 'τοποθετούμε τα sprites
81 i=0
89 ----- ΚΥΡΙΟΣ ΚΥΚΛΟΣ ΠΑΙΧΝΙΔΙΟΥ (ΚΥΚΛΟΣ
ΠΑΙΧΝΙΔΙΟΥ) -----
90 i=i+1:gosub 100
92 εάν i=3 τότε i=0
93 |AUTOALL
94 |PRINTSPALL,1,0: ' κινεί και εκτυπώνει τους 3 στρατιώτες
95 goto 90
96 ----- ΤΕΛΟΣ ΤΟΥ ΚΥΚΛΟΥ ΠΑΙΧΝΙΔΙΟΥ -----
-----
99 ----- ρουτίνα στρατιώτη -----
100 |PEEK,27003+i*16,@x%: x(i)=x%
101 ΑΝ διεύθυνση(i)=0 ΤΟΤΕ ΑΝ x(i)>=xmax(i) ΤΟΤΕ
διεύθυνση(i)=1:|SETUPSP,i,7,10: |SETUPSP,i,6,-1 ELSE return
110 ΑΝ x(i)<=xmin(i) ΤΟΤΕ address(i)=0:|SETUPSP,i,7,9 :
|SETUPSP,i,6,1
120 επιστροφή

```

Κάθε στρατιώτης έχει τη δική του λογική, αλλά εκτελούμε μόνο έναν σε κάθε κύκλο παιχνιδιού, κάνοντας τον κύκλο παιχνιδιού πολύ πιο ελαφρύ.

Ο μόνος περιορισμός είναι ότι με την εκτέλεση της λογικής κάθε στρατιώτη μία από τις 3 φορές, η συντεταγμένη θα μπορούσε να υπερβεί το όριο που έχουμε θέσει για δύο κύκλους. Αυτό μας κάνει να πρέπει να είμαστε πιο προσεκτικοί όταν θέτουμε το όριο, φροντίζοντας όταν το τρέχουμε να μην εισβάλει ποτέ και να μην σβήσει έναν τοίχο του λαβύρινθου της οθόνης μας, για παράδειγμα. Θα προσπαθήσω να εξηγήσω αυτό το πρόβλημα με μεγαλύτερη ακρίβεια:

Ας υποθέσουμε ότι έχουμε 8 sprites και το sprite μας κινείται σε κάθε κύκλο, αλλά εκτελούμε τη λογική του μόνο μία από τις 8 φορές. Φανταστείτε ένα sprite που βρίσκεται στη θέση x=20 και θέλουμε να μετακινθεί στη θέση x=30 και να γυρίσει. Ας θεωρήσουμε ότι το sprite έχει αυτόματη κίνηση με Vx=1. Σε αυτή την περίπτωση θα ελέγξουμε τη θέση του όταν x=20, x=28, x=36. Όταν φτάσουμε στο 36 θα συνειδητοποιήσουμε ότι έχουμε πάει πολύ μακριά!!! και θα αλλάξουμε την ταχύτητα του sprite σε Vx=-1.

Όπως μπορείτε να δείτε, ο έλεγχος των ορίων της τροχιάς δεν είναι ακριβής, εκτός αν λάβουμε υπόψη αυτή την περίσταση και θέσουμε το όριο σε κάτι που μπορούμε να ελέγξουμε, το οποίο θα είναι Xfinal = Xinitial + n*8.

Αυτός ο περιορισμός είναι ελάχιστος σε σύγκριση με το πλεονέκτημα της μετακίνησης πολλών sprites σε υψηλή ταχύτητα. Με λίγη πονηριά μπορούμε ακόμη και να εκτελέσουμε τη λογική λιγότερες φορές, έτσι ώστε μόνο κάθε δεύτερο κύκλο να εκτελείται κάποιο είδος λογικής του εχθρού.

11.3.4 Μπλόκο" μετακίνησης μοίρας

Αν θέλετε απλώς να μετακινήσετε μια μοίρα προς μια κατεύθυνση κάθε φορά, θα λειτουργήσει οποιαδήποτε από τις δύο παρακάτω συναρτήσεις από τη βιβλιοθήκη 8BP:

- Αν χρησιμοποιήσετε το **|AUTOALL** θα πρέπει να επιταχύνετε αυτόματα τα sprites προς την κατεύθυνση που θέλετε (στο Vx, στο Vy ή και στα δύο) και φυσικά να ορίσετε το bit 4 του status byte. Η εντολή AUTOALL έχει μια προαιρετική παράμετρο για την εσωτερική κλήση της εντολής **|ROUTEALL** πριν από τη μετακίνηση των sprites.

- Αν χρησιμοποιήσετε το **|MOVERALL** θα πρέπει να ορίσετε το bit 5 του byte κατάστασης στα sprites που πρόκειται να μετακινήσετε. Αυτή η εντολή απαιτεί ως παραμέτρους πόση σχετική μετακίνηση σε Y και X θέλετε.

Με αυτόν τον τρόπο, με μία μόνο εντολή μετακινείτε πολλά sprites ταυτόχρονα. Στην περίπτωση που κάθε ένα από τα sprites σας πρέπει να κινείται ανεξάρτητα και με ανεξάρτητη λογική, όπως συμβαίνει σε παιχνίδια όπως το "pacman", θα πρέπει να είστε πιο πονηροί, όπως θα σας πω στη συνέχεια.

11.3.5 Μαζική λογική τεχνική σε παιχνίδια τύπου "pacman"

Αν έχετε πολλούς εχθρούς και πρέπει να παίρνουν αποφάσεις σε κάθε διακλάδωση ενός λαβύρινθου, δεν είναι καλή στρατηγική να εκτελείτε εναλλάξ τη λογική του εχθρού σε κάθε κύκλο. Το καλύτερο είναι να εκτελείτε τη λογική όταν αυτή η λογική πρέπει να λάβει μια απόφαση. Στα παιχνίδια τύπου pacman αυτό συμβαίνει όταν ένα φάντασμα φτάνει σε μια διακλάδωση όπου μπορεί να πάρει μια νέα κατεύθυνση κίνησης με βάση τη νοημοσύνη του. Αυτό μπορεί να γίνει με ένα απλό "κόλπο". Είναι απλά να τοποθετήσετε τους εχθρούς σε καλά επιλεγμένες θέσεις στην αρχή του παιχνιδιού.

Ας υποθέσουμε ότι έχετε 4 εχθρούς και οι διακλαδώσεις του λαβύρινθου εμφανίζονται σε πολλαπλάσια του 4. Εάν ο πρώτος εχθρός βρίσκεται σε θέση πολλαπλάσια του 4, είναι η σειρά του να εκτελέσει τη λογική του. Ο δεύτερος εχθρός θα εκτελέσει τη λογική της απόφασής του στον επόμενο κύκλο. Εάν δεν βρίσκεται σε θέση διακλάδωσης του λαβύρινθου, δεν μπορεί να αλλάξει πορεία.

Για να "ταιριάξουμε" τη θέση του με ένα πολλαπλάσιο του 4 και να μπορέσουμε έτσι να αποφασίσουμε ποιο μονοπάτι θα ακολουθήσουμε στη διακλάδωση, απλά ξεκινάμε το παιχνίδι με αυτόν τον δεύτερο εχθρό τοποθετημένο σε ένα πολλαπλάσιο του 4 μείον ένα. Λαμβάνοντας υπόψη τις συντεταγμένες που ξεκινούν από το μηδέν, τα πολλαπλάσια του 4 είναι:

Πρώτος εχθρός: Θέση 0 ή 4 ή 8 ή 12 ή 16 ή 20 ή 24 ή XX (στον άξονα x ή y, δεν έχει σημασία) Δευτερος εχθρός: Θέση 1 ή 5 ή 9 ...

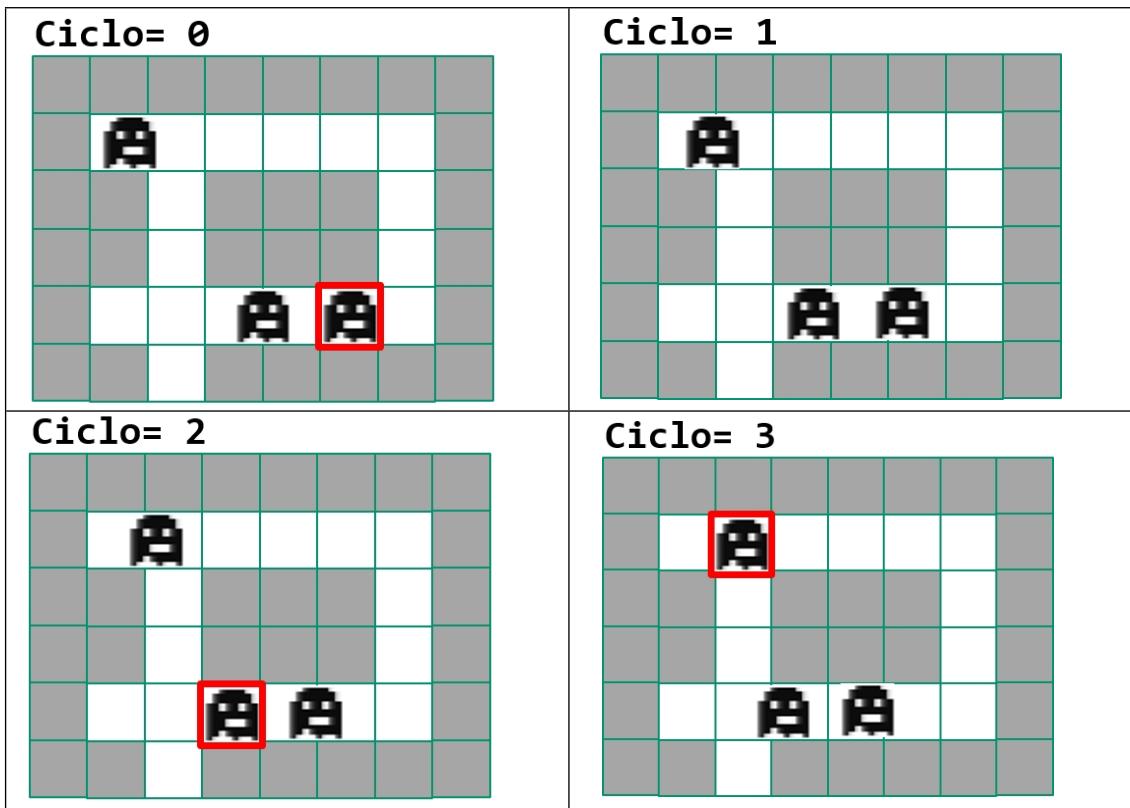
Τρίτος εχθρός: Θέση 2 ή 3 ή 10 ...

Και ούτω καθεξής. Τοποθετείτε τους εχθρούς σας σύμφωνα με αυτόν τον κανόνα:

Θέση = πολλαπλάσιο του 4 - n, όπου n είναι ο αριθμός του sprite

Και κάθε φορά που είναι η σειρά του εχθρού να εκτελέσει τη λογική του, μπορεί να βρεθεί σε μια διακλάδωση του δρόμου. Αν ένα φάντασμα πρέπει να πάρει μια απόφαση τη στιγμή "i", τότε θα πάρει μια απόφαση t=i+4, και η επόμενη απόφαση θα είναι t=i+4+4, και ούτω καθεξής.

Ας δούμε ένα γραφικό παράδειγμα στο οποίο έχω επισημάνει με ένα ορθογώνιο ότι ο εχθρός πρόκειται να πάρει μια απόφαση επειδή βρίσκεται σε μια διακλάδωση. Όπως μπορείτε να δείτε, σε κάθε κύκλο παιχνιδιού εκτελείται μόνο μία λογική διακλάδωσης.



Σχ. 66 Μαζικές λογικές σε παιχνίδια τύπου PACMAN

Όταν έρθει η σειρά σας να εκτελέσετε τη λογική σας, θα πρέπει να ελέγξετε ότι δεν βρίσκεστε σε θέση στη μέση ενός διαδρόμου χωρίς διακλαδώσεις. Θα πρέπει να το ελέγξετε αυτό χρησιμοποιώντας την εντολή **|COLAY**.

Το βιντεοπαιχνίδι "Paco, the man" χρησιμοποιεί αυτή την τεχνική. Στην πραγματικότητα, εκτελεί τη λογική 4 φαντασμάτων, το καθένα σε διαφορετικό κύκλο παιχνιδιού, καθώς και την κατανομή των υπόλοιπων εργασιών στους διάφορους κύκλους. Πρόκειται για ένα θαυμάσιο παράδειγμα μαζικής λογικής, γι' αυτό σας συνιστώ να διαβάσετε το "making of" του βιντεοπαιχνιδιού, το οποίο θα βρείτε στην τεκμηρίωση του 8BP.

	κύκλοι			
Εργασία	0	1		
Ανάγνωση πληκτρολογίου	X			
Ανίχνευση σύγκρου ση με φαντάσματα y sprites αόρατο		X		
Ανίχνευση σύγκρου ση με διάταξη y μετεγκατάσταση εάν O Paco συντρίβεται			X	

Το βιντεοπαιχνίδι "Paco the man" κατανέμει τις εργασίες σε 4 κύκλους παιχνιδιού, επιτρέποντας την επίτευξη σχεδόν 20 FPS σε BASIC.

Ανίχνευση	απ			X	
	ó				
	το				
σημεία					
(καρύδες)					
Λογικό φάντασμα 1	X				
Λογικό φάντασμα 2		X			
Λογική φάντασμα 3			X		
Phantom Logic 4				X	

11.3.6 Μείωση του αριθμού των εντολών στον κύκλο του παιχνιδιού Η μείωση του αριθμού των εντολών που περνάει ο κύκλος του παιχνιδιού είναι απαραίτητη για την επιτάχυνση του προγράμματός σας. Μερικές φορές αυτό σημαίνει ότι το πρόγραμμα θα έχει περισσότερες γραμμές, παρόλο που κατά τέλος εκτελούνται λιγότερες γραμμές σε κάθε κύκλο. Άλλες φορές μπορείτε να εισαγάγετε περιορισμούς "μη ανιχνεύσιμα" που επιταχύνουν το παιχνίδι σας χωρίς ο παίκτης να αντιληφθεί τη διαφορά. Ας ρίξουμε μια ματιά σε μερικά παραδείγματα

11.3.6.1 Διαχείριση πληκτρολογίου με λιγότερες οδηγίες

Διαχειριστείτε το πληκτρολόγιο (και γενικά αυτό ισχύει για οτιδήποτε κάνετε) εκτελώντας όσο το δυνατόν λιγότερες εντολές. Ακολουθεί ένα παράδειγμα (αρχικά κακοφτιαγμένο και στη συνέχεια καλοφτιαγμένο), όπου περνάτε το πολύ 4 πράξεις INKEYS με τις αντίστοιχες IF. Τρέξτε το νοερά και θα δείτε τι εννοώ. Είναι πολύ πιο γρήγορο το δεύτερο

Κακό παράδειγμα (χειρότερη περίπτωση = 8 εκτελέσεις "IF INKEY"):

```
1000 rem αποτελεσματική ρουτίνα πληκτρολογίου
1010 AN INKEY(27)=0 και INKEY(67)=0 TOTE <οδηγίες>:RETURN
1020 AN INKEY(27)=0 και INKEY(69)=0 TOTE <οδηγίες>:RETURN
1030 AN INKEY(34)=0 και INKEY(67)=0 TOTE <οδηγίες>:RETURN
1040 AN INKEY(34)=0 και INKEY(69)=0 TOTE <οδηγίες>:RETURN
1050 IF INKEY(27)=0 THEN <instructions>:RETURN
1060 IF INKEY(34)=0 THEN <instructions>:RETURN
1070 IF INKEY(67)=0 THEN <instructions>:RETURN
1080 IF INKEY(69)=0 THEN <instructions>:RETURN
1080 IF INKEY(69)=0 THEN <instructions>:RETURN
```

Ακολουθεί το παράδειγμα μιας ανάγνωσης πληκτρολόγησης, αλλά καλά εκτελεσμένης (με χειρότερη περίπτωση = 4 εκτελέσεις "IF INKEY"). Επίσης, έχει ληφθεί υπόψη ότι, σε ένα IF, οι μη μηδενικές εκφράσεις είναι ΑΛΗΘΙΝΕΣ. Οι εντολές που πρέπει να εκτελεστούν στο παιχνίδι σας μπορεί να είναι διαφορετικές, αλλά το σχήμα ανάγνωσης του πληκτρολογίου πρέπει να είναι το ίδιο σε περίπτωση χειρισμού διαγωνίων (πάνω και δεξιά ταυτόχρονα, για παράδειγμα). Μπορεί να φαίνεται μεγαλύτερο, αλλά είναι πολύ πιο γρήγορο από το προηγούμενο παράδειγμα.

```
REM αποτελεσματική ρουτίνα
πληκτρολογίου 'μετάβαση στο 1550
αν δεν έχει πατηθεί το "P" 1510 if
inkey(27) THEN 1550: 'πλήκτρο P 1520 if
inkey(67) THEN 1530: 'πλήκτρο Q

1525 <Οδηγίες σε περίπτωση που έχετε πατήσει ταυτόχρονα τα πλήκτρα
"P" και "Q"
time>:RETURN
1530 if inkey(69) THEN 1540:'A' key

1535 <Οδηγίες σε περίπτωση που έχετε πατήσει ταυτόχρονα τα πλήκτρα
"P" και "A"
time>:RETURN
1540 <Οδηγίες σε περίπτωση που έχετε πατήσει μόνο το "P">:RETURN 1550

if inkey(34) THEN 1590:'O' key
```

1560 if INKEY(67) THEN 1570: πλήκτρο 'Q'

1565 <Οδηγίες σε περίπτωση που έχετε πατήσει ταυτόχρονα τα πλήκτρα "O" και "Q" time>:RETURN

1570 if INKEY(69) THEN 1580: 'κλειδί A

1575 <Οδηγίες αν έχετε πατήσει ταυτόχρονα τα πλήκτρα "O" και "A">: RETURN

1580 <Οδηγίες αν έχετε πατήσει μόνο το "O">:RETURN

1590 EAN INKEY(67) TOTE 1600: πλήκτρο "Q".

1595 <οδηγίες αν έχει πατηθεί το "Q">: RETURN

1600 AN INKEY(69) TOTE επιστροφή: "A" κλειδί

1610 <Οδηγίες εάν έχει πατηθεί μόνο το "A">: RETURN

Ένα άλλο πράγμα που πρέπει να κάνετε για να επιταχύνετε το παιχνίδι σας είναι να χρησιμοποιείτε μια περιοδική εργασία για να σαρώνετε "δευτερεύοντα" πλήκτρα, όπως πλήκτρα για την ενεργοποίηση/απενεργοποίηση της μουσικής, πλήκτρα για την εναλλαγή σε ένα μενού ή την εμφάνιση κάτι ειδικού, κ.λπ. Αυτά είναι πλήκτρα που μπορείτε να σαρώνετε περιοδικά και όχι σε κάθε κύκλο. Ωστόσο, πρέπει να λάβετε υπόψη σας πόσο κοστίζει η σάρωσή τους, η οποία δεν είναι μεγάλη (1 ms).

10 if inkey(47) then 30: ' αυτό κοστίζει 1,0 ms

20 <Οδηγίες αν πατήσετε το πλήκτρο 47>.

30 rem φτάνετε εδώ αν δεν έχετε κάνει κλικ σε αυτό

Η ανάλυση μιας μεταβλητής με AND για μια εργασία που θα συμβαίνει κάθε (για παράδειγμα) 4 κύκλους κοστίζει 1,18 ms, οπότε θα μιας κοστίσει

1,18 ms x 4 κύκλοι (αξιολόγηση κύκλων) + 1,0ms (**inkey**) =5,72 ms

αν αντί να το κάνουμε αυτό, εκτελούσαμε το **inkey** σε κάθε κύκλο, θα είχαμε ξοδέψει μόνο 4ms, επομένως, η σάρωση ορισμένων πλήκτρων με βάση τον κύκλο αναπαραγωγής έχει νόημα μόνο αν πρόκειται τουλάχιστον να αποφύγουμε τη σάρωση σε ορισμένους κύκλους δύο πλήκτρων.

Ας υποθέσουμε το ακόλουθο πρόγραμμα:

10 if cycle and 3 then 50: ' κοστίζει 1,18 ms

20 if inkey(47) ... ' αυτό κοστίζει 1 ms

30 if inkey(35) ...' αυτό κοστίζει 1 ms

50 <περισσότερες οδηγίες>.

Ως έχει, το πρόγραμμα αξιολογεί τα πλήκτρα 47 και 35 κάθε 4ο κύκλο. Όταν τα αξιολογεί δαπανά $1,18 + 1 + 1 + 1 = 3,8$ ms ενώ όταν δεν τα αξιολογεί δαπανά 1,18 ms. Επομένως, ο χρόνος που δαπανάται σε 4 κύκλους είναι Χρόνος $3 * 1,18 + 3,8 = 6,72$ ms

Ενώ αν είχαμε αξιολογήσει τα πλήκτρα κάθε κύκλο, θα είχαμε δαπανήσει $4 * 2$ ms= 8 ms. Επομένως, υπάρχει εξοικονόμηση $8 - 6,7 = 1,3$ ms για κάθε 4 κύκλους, ή περίπου 0,3 ms ανά κύκλο παιχνιδιού.

Μία από τις επιλογές που έχετε, ανάλογα με τον τύπο του παιχνιδιού, είναι να σαρώνετε

ορισμένα πλήκτρα σε ζυγούς κύκλους και τα υπόλοιπα σε περιττούς κύκλους. Για παράδειγμα, σε ένα παιχνίδι που χρησιμοποιεί πλήκτρα QAOP, μπορείτε να σαρώσετε το QA σε ζυγούς κύκλους και το OP σε περιττούς κύκλους ή το αντίστροφο.

Με αυτόν τον τρόπο μπορείτε να επιτύχετε μεγαλύτερη ταχύτητα με έναν περιορισμό που ο παίκτης είναι απίθανο να παρατηρήσει. Αυτό είναι το είδος του περιορισμού που αξίζει μερικές φορές να εισαχθεί, αλλά εξαρτάται από το παιχνίδι.

11.3.6.2 Αποφύγετε να περνάτε από περιττά FI

Θα εξετάσουμε δύο τρόπους για να το κάνουμε αυτό:

```
10 IF A=1 THEN <οδηγίες όταν A=1> 10 IF A=1 THEN <οδηγίες όταν A=1> 10  
IF A=1 THEN <οδηγίες όταν A=1> 10 IF A=1 THEN <οδηγίες όταν A=1>  
20 IF A=2 THEN <οδηγίες όταν A=2> 20 IF A=2 THEN <οδηγίες όταν A=2> 20  
IF A=2 THEN <οδηγίες όταν A=2>  
30 AN A=3 TOTE <οδηγίες όταν A=3>  
40 <περισσότερες οδηγίες>
```

Αν μπορείτε να αποφύγετε να περάσετε από ένα IF εισάγοντας ένα GOTO, είναι πάντα προτιμότερο. Η GOTO είναι ένας μεγάλος σύμμαχος της τεχνικής της μαζικής λογικής.

```
10 EAN A=1 TOTE <εντολές όταν A=1> : GOTO 40  
20 IF A=2 THEN <οδηγίες όταν A=2> : GOTO 40  
30 <οδηγίες όταν A=3> : rem αν το A δεν είναι ούτε 1 ούτε 2 τότε είναι  
3  
40 <περισσότερα οδηγίες>
```

Ένας άλλος τρόπος για να γίνει αυτό είναι η χρήση της εντολής ON <μεταβλητή> GOTO ή της εντολής ON <μεταβλητή> GOSUB.

Όπως έχω παρουσιάσει στον πίνακα του 11.1, μπορείτε να εξοικονομήσετε περισσότερα από 1ms με τη χρήση του ON GOTO.

```
10 σε A GOTO 30,40,50  
20 goto 60: rem φτάνουμε εδώ αν A=4  
30 rem φτάνουμε εδώ αν A=1  
35 goto 60  
40 rem φτάνουμε εδώ αν A=2  
45 επιλέξτε 60  
50 rem φτάνουμε εδώ αν A=3  
60 rem εδώ η λογική συνεχίζεται
```

11.3.6.3 Αντικαθιστά αλγόριθμους με προ-υπολογισμούς

Σκεφτείτε μια μπάλα που αναπηδά. Αντί να χρησιμοποιήσετε τις εξισώσεις της επιταχυνόμενης κίνησης, κατασκευάστε μια διαδρομή που κινεί ένα sprite προς τα κάτω με μεγαλύτερη αύξηση της συντεταγμένης Y σε κάθε βήμα και στη συνέχεια προς τα πάνω με όλο και μικρότερη αύξηση καθώς χτυπάει στο έδαφος. Δεν υπάρχουν πολύπλοκες εξισώσεις που πρέπει να εκτελεστούν και παρόλα αυτά το οπτικό αποτέλεσμα είναι το ίδιο. Με αυτόν τον τρόπο έκανα τα άλματα στο παιχνίδι "Φρέσκα φρούτα και λαχανικά". Είναι δυνατό να προγραμματιστεί με αυτόν τον τρόπο επειδή μέσα στο παιχνίδι το σύμπαν είναι "ντετερμινιστικό". Δηλαδή, μπορείτε να προβλέψετε σε κάθε χρονική στιγμή τι πρόκειται να συμβεί, ανεξάρτητα από το πόσο πολύπλοκες είναι οι εξισώσεις που διέπουν το άλμα ενός χαρακτήρα ή την κίνηση μιας ομάδας.

Σε παιχνίδια όπου η λογική του εχθρού απαιτεί τη χρήση κάποιας συνάρτησης υπολογισμού (όπως το συνημίτονο), υπολογίστε τα πάντα εκ των προτέρων και αποθηκεύστε τα σε έναν πίνακα που θα χρησιμοποιείτε κατά τη διάρκεια της εκτέλεσης της λογικής. Ο υπολογισμός κατά τη διάρκεια της λογικής του παιχνιδιού είναι απαγορευτικός από πλευράς κόστους.

Η σύνθετη λογική είναι αργή λογική. Αν θέλετε να φτιάξετε κάτι πολύπλοκο, μια πολύπλοκη τροχιά, έναν μηχανισμό τεχνητής νοημοσύνης... μην το κάνετε, προσπαθήστε να το "προσομοιώσετε" με ένα απλούστερο μοντέλο συμπεριφοράς, αλλά να παράγετε το ίδιο οπτικό αποτέλεσμα. Για παράδειγμα, ένα φάντασμα που είναι έξυπνο και σας κυνηγάει, αντί να το βάλετε να παίρνει έξυπνες αποφάσεις, βάλτε το να προσπαθεί να πάρει την ίδια κατεύθυνση με τον χαρακτήρα σας, χωρίς καμία λογική. Αν δεν μπορείτε να απλοποιήσετε με αυτόν τον τρόπο, τότε σκεφτείτε ότι ακόμη και η τεχνητή νοημοσύνη μπορεί να γίνει "ντετερμινιστική". Αν ένας εχθρός πάρει μια απόφαση για το πώς θα κινηθεί με βάση τη θέση και την ταχύτητά σας, θα μπορούσαμε να αποθηκεύσουμε το αποτέλεσμα αυτού του βαρέως αλγορίθμου σε έναν πίνακα και να αποφύγουμε όλους τους υπολογισμούς.

10 Rem Vx, Vy, X, Y είναι η ταχύτητα και η θέση του χαρακτήρα μου.
20 rem ας υποθέσουμε ότι έχω υπολογίσει εκ των προτέρων αποφάσεις για 3 ταχύτητες, 10 υποδοχές συντεταγμένων X και 10 υποδοχές Y.
Αυτό είναι λιγότερο από 1KB
30 DIM pursue(3,3,10,100)
40 rem ' φορτώστε τις τιμές στον πίνακα αφού τις υπολογίσετε αργά
50 newaddress=pursue(Vx,Vy,x,y): μηχανισμός rem σε λειτουργία

Το σύμπαν που προγραμματίζετε είναι "ντετερμινιστικό". Όσο πολύπλοκη και αν είναι η συμπεριφορά των εχθρών και των στοιχείων της οθόνης, αν η συμπεριφορά τους δεν εξαρτάται από την αλληλεπίδρασή σας, τότε υπάρχει μια θέση για κάθε δεδομένο πράγμα σε κάθε δεδομένη χρονική στιγμή. Μια θέση που θα μπορούσε να υπολογιστεί εκ των προτέρων για να αποφευχθεί κάθε πολύπλοκος αλγόριθμος συμπεριφοράς και το αποτέλεσμα θα ήταν το ίδιο.

11.3.6.4 Μην εκτελείτε σχόλια

Αφαιρέστε τυχόν σχόλια στη λογική του παιχνιδιού και αν αφήσετε σχόλια που είναι REM (πιο γρήγορα), μη χρησιμοποιείτε εισαγωγικά. Αν χρησιμοποιήσετε το εισαγωγικό είναι για να εξοικονομήσετε 2 bytes μνήμης και είναι κατάλληλο για να σχολιάσετε το υπόλοιπο πρόγραμμα (αρχικοποίσεις και τέτοια). Αν θέλετε να σχολιάσετε τμήματα της λογικής μπορείτε να κάνετε τα εξής:

Εάν x>23 gosub 500
 ...
499 rem δεν περνάει κατά μήκος αυτής της γραμμής και έτσι σχολιάζω αυτή τη ρουτίνα.
500 εάν x > 50 TOTE ...
 ...
550 ΕΠΙΣΤΡΟΦΗ

Κάθε σχόλιο που εκτελείτε καταναλώνει 0,20 ms και η αποθήκευση της εκτέλεσής του είναι πολύ εύκολη χωρίς να αφήνετε σχόλια. Υπάρχουν φορές που μπορείτε να βάζετε σχόλια σε γραμμές εφόσον υπάρχουν άλματα (GOTO και GOSUB/RETURN) χωρίς να φοβάστε ότι θα χάσετε χρόνο, ας δούμε μερικά παραδείγματα:

10 goto 50 : rem αυτό το σχόλιο δεν είναι χρονοβόρο
20 gosub 200: rem αυτό το σχόλιο δεν είναι χρονοβόρο
200 επιστροφή: rem αυτό το σχόλιο δεν είναι χρονοβόρο

11.3.6.5 Μόνο ένα sprite πεθαίνει σε κάθε κύκλο

Η εντολή 8BP |COLSPALL έχει σχεδιαστεί με γνώμονα τη "μαζική λογική". Αυτό

σημαίνει ότι δυνητικά ανιχνεύει τη σύγκρουση όλων των sprites, αλλά μόλις ανιχνεύσει μια σύγκρουση επιστρέφει υποδεικνύοντας ποιο sprite είναι ο συγκρουόμενος και ποιο είναι το sprite που συγκρούστηκε. Σε αυτό το σημείο μπορείτε να παρακάμψετε το συγκρουόμενο sprite (απενεργοποιώντας την ικανότητά του να συγκρούεται στο bit 1 του status byte του) και να εκτελέσετε ξανά την εντολή, μέχρι να μην υπάρχουν άλλες συγκρούσεις (επιστρέφει ένα 32 για το συγκρουόμενο και το συγκρουόμενο). Ωστόσο,

Είναι πιο αποτελεσματικό να μην ενεργοποιήσετε ξανά την εντολή μέχρι τον επόμενο κύκλο παιχνιδιού. Αυτό σημαίνει ότι μόνο ένας εχθρός μπορεί να πεθάνει σε κάθε καρέ, αλλά είναι ένας ανεπαίσθητος περιορισμός που θα επιταχύνει σημαντικά τα παιχνίδια σας.

11.3.7 Διαδρομές που επιταχύνουν το παιχνίδι με χειρισμό της κατάστασης

Μπορείτε να φτιάξετε μονοπάτια που ενεργοποιούν και απενεργοποιούν εναλλάξ τον συγκρουστή ή τη σημαία συγκρουστή στα sprites σας. Ανάλογα με τον τύπο του εχθρού, αυτό μπορεί να σας δώσει επιπλέον ταχύτητα στην εντολή σύγκρουσης. Οι διαδρομές εξηγούνται σε επόμενο κεφάλαιο, οπότε πριν διαβάσετε αυτό, καλό είναι να εξοικειωθείτε με αυτές.

Για παράδειγμα, αν έχετε 8 εχθρούς στην οθόνη, μπορείτε να κάνετε 4 συγκρουόμενους εχθρούς στους ζυγούς κύκλους και 4 συγκρουόμενους εχθρούς στους περιττούς κύκλους:

ΔΙΑΔΡΟΜΗ1,

db 1,0,1
db 255,128+8+2+1,0 ; αλλαγή της κατάστασης σε συγκρουόμενη db 1,0,1
db 255,128+8+1,0 ; αλλαγή κατάστασης σε μη συγκρουόμενη db 0

Αυτή η διαδρομή αλλάζει την κατάσταση του sprite σας, ενώ το μετακινεί προς τα δεξιά. Εάν σε ένα sprite έχει ανατεθεί αυτό το μονοπάτι, θα κινείται προς τα δεξιά και θα είναι εναλλάξ συγκρουόμενο και μη συγκρουόμενο.

Μπορείτε να αναθέσετε τη διαδρομή σε 8 sprites και στη συνέχεια να εκτελέσετε την ακόλουθη εντολή σε 4 από τα sprites:

|ROUTESP, <sprite_id>, 1

Με αυτό, η εντολή σύγκρουσης **|COLSPALL** θα είναι ταχύτερη, καθώς θα πρέπει να ανιχνεύει συγκρούσεις μόνο με 4 συγκρουόμενους εχθρούς σε κάθε καρέ. Αυτό το κόλπο επιταχύνει λίγο το παιχνίδι σας και μαζί με άλλα κόλπα μπορείτε τελικά να φτάσετε στα FPS που χρειάζεστε.

Το ίδιο κόλπο σας επιτρέπει να ενεργοποιείτε και να απενεργοποιείτε τη σημαία εκτύπωσης (bit 0 της κατάστασης) και σε ένα μονοπάτι όπου ο εχθρός περνάει κάποιο καρέ χωρίς να κινείται, μπορείτε να απενεργοποιήσετε τη σημαία εκτύπωσης μέσω του μονοπατιού και έτσι να επιταχύνετε την εντολή PRINTSPALL.

11.3.8 Δρομολόγηση sprites με "μαζική λογική".

Για να μετακινήσετε sprites μέσα από μονοπάτια υπάρχει μια εντολή που ονομάζεται **|ROUTEALL** που το κάνει πολύ αποτελεσματικά, αλλά ως **άσκηση για την κατανόηση της φιλοσοφίας της λογικής του όγκου είναι πολύ ενδιαφέρον να μελετήσετε αυτή τη δύσκολη αλλά εμβληματική περίπτωση της λογικής του όγκου**. Για τον προγραμματισμό του βιντεοπαιχνιδιού "Anunnaki" χρησιμοποίησα την τεχνική που θα περιγράψω παρακάτω, καθώς δεν είχα προγραμματίσει ακόμα τη δυνατότητα δρομολόγησης sprites στο 8BP. Βασικά χρησιμοποίησα μαζική λογική στη δρομολόγηση των εχθρικών πλοιών.

Τα πλοία θα περνούν ένα προς ένα από μια σειρά "κόμβων ελέγχου", που είναι σημεία στο χώρο όπου πρέπει να αλλάξουν την κατεύθυνσή τους, η οποία ορίζεται από τις

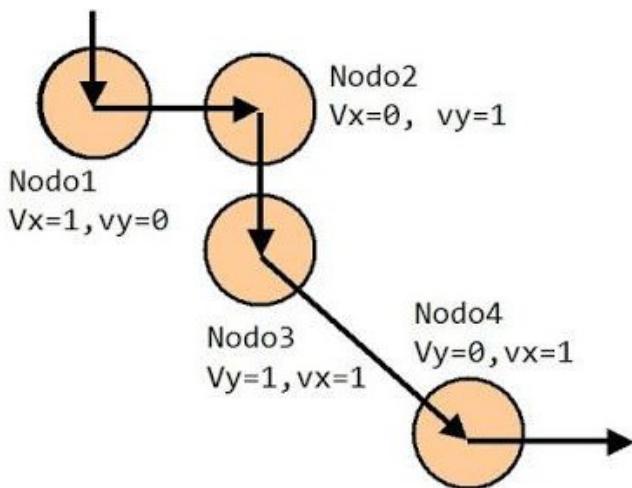
ταχύτητές τους σε X και Y, δηλαδή (Vx, Vy) .

Ένας τρόπος για να ελέγξουμε ότι τα 8 πλοία αλλάζουν κατεύθυνση σε αυτές τις θέσεις θα ήταν να συγκρίνουμε τις συντεταγμένες X, Y τους με αυτές κάθε ενός από τους κόμβους ελέγχου και αν συμπίπτουν με κάποιον από αυτούς, τότε εφαρμόζουμε τις νέες ταχύτητες που σχετίζονται με τις

αλλαγή σε αυτόν τον κόμβο. Εφόσον μιλάμε για 2 συντεταγμένες, 8 πλοία και 4 κόμβους, εξετάζουμε:

2 x 8 x 4 = 64 έλεγχοι σε κάθε πλαίσιο

Αυτό δεν είναι εφικτό αν θέλουμε ταχύτητα από τη BASIC, καθώς δεν είναι μια υπολογιστικά αποδοτική στρατηγική. Εφόσον έχουμε να κάνουμε με ένα "ντετερινιστικό" σενάριο, μπορούμε να είμαστε σίγουροι σε κάθε χρονική στιγμή για το πού θα βρίσκεται κάθε ένα από τα πλοία και επομένως, αντί να ελέγχουμε στο χώρο, μπορούμε να επικεντρωθούμε μόνο στη χρονική συντεταγμένη (η οποία είναι ο αριθμός πλαισίου του παιχνιδιού ή ο λεγόμενος αριθμός "κύκλου παιχνιδιού"). Μην εξετάζετε το χρόνο σε δευτερόλεπτα, αλλά σε καρέ.



Σχ. 67 Καθορισμένη τροχιά με "κόμβους ελέγχου".

Εφόσον γνωρίζουμε την ταχύτητα με την οποία κινούνται τα πλοία, μπορούμε να γνωρίζουμε πότε το πρώτο πλοίο θα περάσει από τον πρώτο κόμβο. Θα ονομάσουμε αυτή τη στιγμή $t(1)$. Θα υποθέσουμε επίσης ότι λόγω της απόστασης μεταξύ των πλοίων, το δεύτερο από τα πλοία θα περάσει τον κόμβο τη στιγμή $t(1)+10$. Το τρίτο τη στιγμή $t(1)+20$ και το όγδοο τη στιγμή $t(1)+70$. Αντί να χρησιμοποιούμε πλαίσια ως μονάδες χρόνου, ας χρησιμοποιήσουμε δεκάδες πλαίσια: σε αυτή την περίπτωση οι χρονικές στιγμές θα είναι $t(1), (1)+1, t(1)+2, \dots$, κ.λπ.

Γνωρίζοντας αυτό μπορούμε να ελέγξουμε το χρόνο με δύο μεταβλητές: η μία θα μετράει τις δεκάδες (i) και η άλλη τις μονάδες (j). Για να ελέγξουμε την αλλαγή των 8 πλοίων στον πρώτο κόμβο μπορούμε να γράψουμε:

j=j+1: ΑΝ j=10 ΤΟΤΕ j=0: i=i+1: ΑΝ i>t(1) ΚΑΙ i<t(1) +8 ΤΟΤΕ [ενημερώνει την ταχύτητα του πλοίου i-t(1) με τις τιμές ταχύτητας του κόμβου 1].

Όπως βλέπουμε, με μία μόνο γραμμή μπορούμε να αλλάξουμε τις ταχύτητες κάθε πλοίου καθώς περνούν από τον κόμβο 1. Κάθε φορά που το " j " γίνεται μηδέν, ανξάνουμε τη μεταβλητή " i " και ενημερώνουμε ένα από τα πλοία. Κατά τη διάρκεια των πρώτων 80 χρονικών στιγμών (8 σε δεκάδες καρέ) κάθε ένα από τα 8 πλοία ενημερώνεται, ακριβώς όπως περνούν από τον κόμβο ελέγχου, δηλαδή τη στιγμή $t(1)$ ενημερώνεται το Sprite 0, τη στιγμή $(t1)+1$ ενημερώνεται το Sprite 1, τη στιγμή $t(1)+2$ ενημερώνεται το Sprite 2 κ.ο.κ.

Ο αριθμός Sprite που εμφανίζεται στη γραμμή είναι $i-t(1)$, οπότε αν $t(1) = 1$ θα είναι το πλαίσιο 40, ($t(1)=4$) τότε όταν το "i" είναι 4 θα αρχίσει να ενημερώνει το Sprite 0, και όταν το "i" είναι 11 θα ενημερώσει το Sprite 7 (8 πλοία συνολικά).

Τώρα ας εφαρμόσουμε το ίδιο και στους 4 κόμβους. Θα μπορούσαμε να εκτελέσουμε 4 ελέγχους αντί για έναν, αλλά θα ήταν αναποτελεσματικό. Επίσης, αν είχαμε πολλούς κόμβους αυτό θα σήμαινε πολλούς ελέγχους. Μπορούμε να το κάνουμε με έναν μόνο, λαμβάνοντας υπόψη ότι το πρώτο πλοίο περνάει από έναν κόμβο τη στιγμή $t(n)$ και το όγδοο πλοίο περνάει από αυτόν τον κόμβο τη στιγμή $t(n)+7$.

Όταν το πρώτο πλοίο περνάει από τον πρώτο κόμβο, είναι λογικό να σκεφτείτε να αρχίσετε να ελέγχετε τον κόμβο 2, αλλά όχι τον κόμβο 3 ή τον κόμβο 4.

Όσον αφορά τον μικρότερο κόμβο, μπορούμε να υποθέσουμε ότι, ακόμη και αν έχουμε 20 κόμβους, αυτοί απέχουν αρκετά μεταξύ τους ώστε να μην υπάρχουν πλοία που διασχίζουν περισσότερους από 3 κόμβους ταυτόχρονα (θα το υποθέσουμε αυτό και θα χρησιμοποιήσουμε το "3" ως παράμετρο). Επομένως, ο μικρότερος κόμβος που πρέπει να ελέγχουμε είναι ο μεγαλύτερος - 3. Θα ονομάσουμε τον μικρότερο κόμβο "nmin" και τον μεγαλύτερο "nmax" ($nmin = nmax - 3$). Σε περίπτωση που θέλουμε να έχουμε πλήρη ελευθερία να ορίσουμε οποιαδήποτε τροχιά, το $nmin$ πρέπει να είναι το $nmax$ μείον τον αριθμό των πλοίων στη σειρά.

```
10 j=j+1: AN j=10 TOTE j=0: i=i+1: n=nmax
20 IF n<nmin THEN 50: δεν υπάρχουν άλλα πλοία προς ενημέρωση
30 EAN i>=t(n) KAI i< t(n)+8 TOTE [ενημέρωση του πλοίου i-t(n)
με τις ταχύτητες του κόμβου n]:EAN i-t(n)=0 TOTE nmax=nmax+1:
nmin=nmax-3
40 n=n-1
```

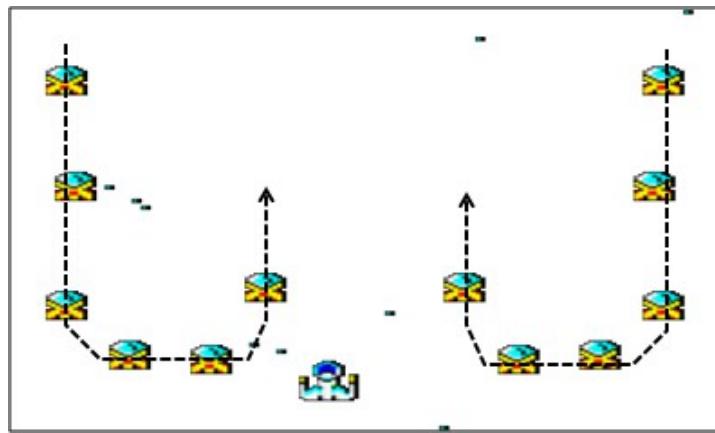
50 ' plus οδηγίες παιχνιδιού

Όπως μπορείτε να δείτε, όταν η χρονική δεκαετία (μεταβλητή "i") αυξάνεται κατά 1, αρχίζει να ελέγχει αν υπάρχει κάποιο πλοίο σε έναν από τους κόμβους από "nmax" έως "nmin", ενημερώνοντας μόνο ένα πλοίο σε κάθε καρέ. Εάν το πλοίο που ενημερώνεται είναι μηδέν, τότε ο μέγιστος κόμβος αυξάνεται, καθώς το πλοίο αυτό βρίσκεται καθ' οδόν προς τον επόμενο κόμβο.

Για το επόμενο καρέ, ο αριθμός των κόμβων μειώνεται (εντολή $n = n-1$), έτσι ώστε να ελέγχουμε αν υπάρχει πλοίο στον προηγούμενο κόμβο, και ούτω καθεξής μέχρι το $nmin$. Πάντα, όμως, ελέγχουμε μόνο ένα πλοίο σε κάθε καρέ.

Εν ολίγοις, μετατρέψαμε 64 ελέγχους σε μόλις 1, χρησιμοποιώντας "Mass Logic". Και αν το μονοπάτι είχε 40 αντί για 4 κόμβους, θα είχαμε μετατρέψει 640 λειτουργίες σε μία!

Το βιντεοπαιχνίδι "Annunaki" χρησιμοποιεί αυτή την τεχνική για να διαχειριστεί τις τροχιές δύο συμμετρικών σειρών των 6 πλοίων η κάθε μία. Είναι περίπλοκο, αλλά όπως μπορείτε να δείτε, από τη BASIC μπορείτε να πάρετε τον έλεγχο 12 πλοίων και να τα κάνετε να κινούνται σε ιδιόρρυθμες τροχιές, χρησιμοποιώντας περισσότερη ευφυΐα παρά ισχύ, χάρη στην τεχνική της μαζικής λογικής.



Σχ. 68 Λύο σειρές με μαζική λογική

12 Πολύπλοκες τροχιές: εντολή ROUTEALL

Πρόκειται για μια "προηγμένη" εντολή που είναι διαθέσιμη από την έκδοση V25 της βιβλιοθήκης 8BP. Απλοποιεί πολύ τον προγραμματισμό, επειδή μπορείτε να ορίσετε μια διαδρομή και να κάνετε ένα sprite να την διανύσει βήμα προς βήμα χρησιμοποιώντας την εντολή ROUTEALL.

Πρώτα, πρέπει να δημιουργήσετε μια διαδρομή. Για το σκοπό αυτό πρέπει να την επεξεργαστείτε στο αρχείο routes_yourgame.asm.

Κάθε διαδρομή έχει απροσδιόριστο αριθμό τμημάτων (αν και το μέγιστο μήκος μιας διαδρομής είναι 255 bytes) και κάθε τμήμα έχει τρεις παραμέτρους:

- Πόσα βήματα θα κάνουμε σε αυτό το τμήμα (μεταξύ 1 και 250)
- Ποια ταχύτητα Vy πρέπει να διατηρηθεί κατά τη διάρκεια του τμήματος ($-127 \leq Vy \leq 127$)
- Ποια ταχύτητα Vx πρέπει να διατηρηθεί κατά τη διάρκεια του τμήματος ($-127 \leq Vx \leq 127$)

Καθώς μια διαδρομή μπορεί να έχει μήκος το πολύ 255 bytes και ένα τμήμα καταλαμβάνει 3 bytes, μια διαδρομή μπορεί να έχει το πολύ 84 τμήματα.

Στο τέλος της προδιαγραφής του τμήματος πρέπει να βάλουμε ένα μηδέν για να δηλώσουμε ότι το μονοπάτι έχει τερματιστεί και ότι το sprite πρέπει να αρχίσει να διασχίζει το μονοπάτι από την αρχή.

Ας δούμε ένα παράδειγμα:

; ΛΙΣΤΑ ΔΙΑΔΡΟΜΩΝ

;=====

**Βάλτε εδώ τα ονόματα όλων των διαδρομών που
κάνετε ROUTE_LIST**

**dw ROUTE0
dw ROUTE1
dw ROUTE2
dw ROUTE3
dw ROUTE4
dw ROUTE4**

ΟΡΙΣΜΟΣ ΚΑΘΕ ΔΙΑΔΡΟΜΗΣ

;=====

ROUTE0- ένας κύκλος

**;-----
db 5,2,0- πέντε βήματα με Vy=2
db 5,2,-1- πέντε βήματα με Vy=2, Vx=-1
db 5,0,-1
db 5,-2,-1
db 5,-2,0
db 5,-2,1
db 5,0,1
db 5,2,1
db 0**

ROUTE1; αριστερά-δεξιά

**;-----
db 10,0,-1
db 10,0,1
db 0**

ROUTE2; πάνω-κάτω

**;-----
db 10,-2,0
db 10,2,0**

db 0

ROUTE3- ένα

οκτώ

```
;-----db 15,2,0-----  
db 5,2,-1  
db 5,0,-1  
db 25,-2,-1  
db 5,0,-1  
db 5,2,-1  
db 15,2,0  
db 5,2,1  
db 5,0,1  
db 25,-2,1  
db 5,0,1  
db 5,2,1  
db 0
```

ROUTE4; ένας βρόχος και πηγαίνει προς τα αριστερά

```
;-----db 120,0,-1-----  
db 10,-2,-1  
db 20,-2,0  
db 10,-2,1  
db 5,0,1  
db 10,2,1  
db 20,2,0  
db 10,2,-1  
db 80,0,-1  
db 0
```

Τώρα για να χρησιμοποιήσουμε τις διαδρομές από τη BASIC, απλά αναθέτουμε τη διαδρομή σε ένα sprite με την εντολή SETUPSP, υποδεικνύοντας ότι θέλουμε να τροποποιήσουμε την παράμετρο 15, η οποία είναι αυτή που υποδεικνύει τη διαδρομή. Επιπλέον, πρέπει να ενεργοποιήσουμε τη σημαία διαδρομής (bit 7) στο status byte του sprite και θα την θέσουμε μαζί με τη σημαία αυτόματης κίνησης και τις σημαίες animation και print.

```

10 MNHMH 24999
11 ON BREAK GOSUB 280
20 MODE 0:INK 0,0
21 LOCATE 1,20:PRINT "Εντολή |ROUTEALL και ακολουθίες μακροεντολών
εμψύχωσης".
30 CALL &6B78:DEFINT a-z
31 |SETLIMITS,0,80,0,200
40 FOR i=0 TO 31:|SETUPSP,i,0,0,0,0:NEXT
41 x=10
50 FOR i=1 TO 8
51 x=x+20:IF x>=80 THEN x=10:y=y+24
60 |SETUPSP,i,0,143: rem με αυτό ενεργοποιείται η σημαία διαδρομής
70 |SETUPSP,i,7,2:|SETUPSP,i,7,33: rem ακολουθία κινουμένων σχεδίων
μακροεντολών
71 |SETUPSP,i,15,3: εκ νέου εκχώρηση του αριθμού διαδρομής 3
80 |LOCATESP,i,30,70
82 FOR t=1 TO 10:|ROUTEALL:|AUTOALL,0:|PRINTSPALL,1,0:NEXT
91 ΕΠΟΜΕΝΟ
100 |AUTOALL,1:|PRINTSPALL,1,0: rem εδώ το AUTOALL καλεί ήδη το ROUTEALL
120 GOTO 100

```

280 |MUSIC:MODE 1: INK 0,0:OPEN 1

Τα έχουμε όλα. Αυτή η προηγμένη τεχνική θα απλοποιήσει πολύ τον προγραμματισμό σας με θεαματικά αποτελέσματα.



Σχ. 69 διαδρομή σχήματος 8

Οπως είδατε, η εντολή δεν τροποποιεί τις συντεταγμένες των sprites, οπότε πρέπει να μετακινηθούν με |AUTOALL και να εκτυπωθούν (και να εμψυχωθούν) με |PRINTSPALL. Γι' αυτό έχετε μια προαιρετική παράμετρο στην |ROUTEALL, έτσι ώστε η |AUTOALL,1 να καλεί εσωτερικά την |ROUTEALL πριν μετακινήσετε το sprite, γλιτώνοντάς σας από μια κλήση της BASIC που θα διαρκεί πάντα ένα πολύτιμο χιλιοστό του δευτερολέπτου.

12.1 Τοποθετεί ένα Sprite στη μέση μιας διαδρομής : ROUTESP

Αν αναθέσετε πολλά sprites στην ίδια διαδρομή και θέλετε να μπουν όλα σε ένα αρχείο, όπως στο παραπάνω παράδειγμα, τότε πρέπει να βεβαιωθείτε ότι κάθε sprite βρίσκεται σε διαφορετικό σημείο της διαδρομής. Υπάρχουν δύο τρόποι για να το κάνετε αυτό

Η πρώτη συνίσταται στη σταδιακή ανάθεση της διαδρομής στα sprites. Με αυτόν τον τρόπο, το sprite που προηγείται είναι το πρώτο που δρομολογείται και μετά από μερικούς κύκλους παιχνιδιού δρομολογείτε το επόμενο και μετά το επόμενο κ.ο.κ. Σε κάθε κύκλο εκτελείτε την εντολή |AUTOALL,1 και αυτό δρομολογεί τα sprites στα οποία έχει ήδη ανατεθεί διαδρομή, τα οποία θα προηγούνται σε αριθμό βημάτων σε σχέση με τα sprites στα οποία δεν έχει ακόμη ανατεθεί.

Η δεύτερη επιλογή είναι η χρήση της εντολής |ROUTESP.

|ROUTESP, <sprited>, <steps>.

Αυτή η εντολή μετακινεί ένα sprite κατά τον επιθυμητό αριθμό βημάτων (έως 255 βήματα) κατά μήκος της διαδρομής που του έχει ανατεθεί. Στο παράδειγμα έχω επισημάνει με κόκκινο χρώμα την ανάθεση της διαδρομής 8 και τις εντολές |ROUTESP που τοποθετούν κάθε sprite κατά μήκος της διαδρομής.

10 μνήμη 24999

10 ON BREAK GOSUB 12

11 GOTO 20

12 |MUSIC:CALL &BC02:PAPER 0:OPEN 1:MODE 1:END

20 CALL &BC02:DEFINT A-Z:MODE 0

50 CALL &6B78

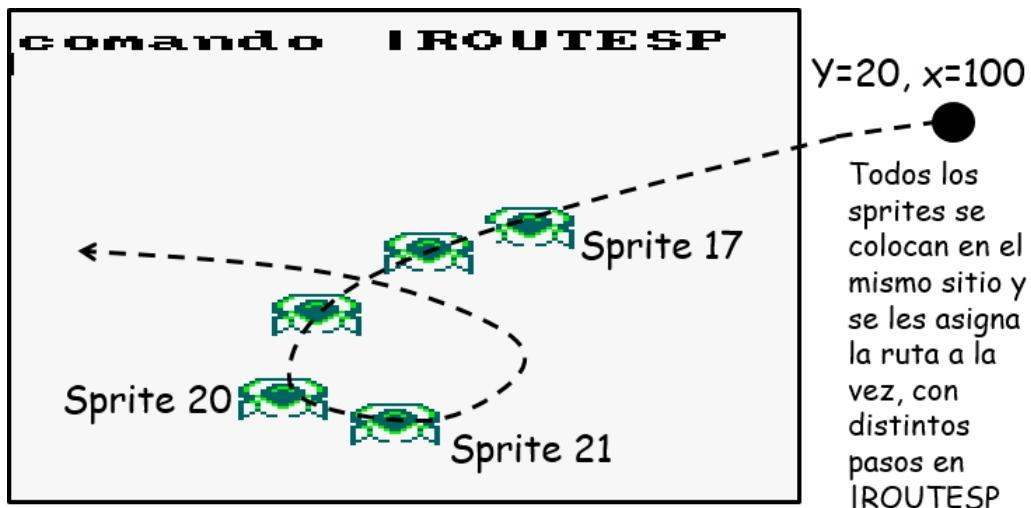
70 * όλα τα σκάφη τοποθετούνται στην ίδια αρχική συντεταγμένη

75 ' καὶ με τὴν ἴδια διαδρομή αλλά με διαφορετικό αρχικό αριθμό βημάτων.

```

76 locate 2,3: Εκτύπωση "εντολή |ROUTEESP "
80 για s=16 έως 21: |SETUPSP,s,9,33: |SETUPSP,s,0,137:
|SETUPSP,s,15,8: |LOCATESP,s,20,100:next
90 s=21: |ROUTEESP,s,40:'κεφαλή πλοίου
100 s=20: |ROUTEESP,s,30
110 s=19: |ROUTEESP,s,20
120 s=18: |ROUTEESP,s,10
130 s=17: |ROUTEESP,s,0
131 |PRINTSPALL,0,0,0,0,0
140 --- κύκλος παιχνιδιού
---
150 |AUTOALL,1: |PRINTSPALL
160 goto 150

```



Σχ. 70 Σπίτια σε σειρά με τη χρήση του ROUTESP

Η διαδρομή 8 ορίζεται στο αρχείο routes_mygame.asm ως εξής:
ΔΙΑΔΡΟΜΗ8,

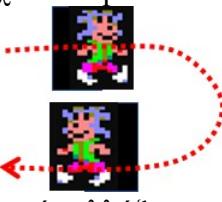
```

db 255, 128+64+32+32+32+8+1,0- αλλαγή κατάστασης
db 70,1,-1
db 10,3,-1
db 5,3,0
db 5,3,1
db 5,0,1
db 20,-3,1
db 5,0,1
db 5,3,1
db 5,3,0
db 10,3,-1
db 5,0,-1
db 20,-3,-1
db 40,-1,-1,-1
;
επανατοποθέτηση db
1,0,115
db 1,-30,0
db 120,0,0
db 120,0,0
db 0

```

12.2 Δημιουργία Σύνθετων διαδρομών

Υπάρχουν 4 λειτουργίες που μπορείτε να χρησιμοποιήσετε στη μέση οποιασδήποτε διαδρομής (**προσοχή, στη μέση, όχι στο τέλος**), χρησιμοποιώντας έναν κωδικό διαφυγής ως τιμή του αριθμού των βημάτων ενός τμήματος:

Κωδικός διαφυγής	Περιγραφή	Παράδειγμα
255	Αλλαγή της κατάστασης του sprite.	DB 255, 3, 0 Το μηδέν στο τέλος είναι συμπλήρωμα.
254	Αλλαγή της ακολουθίας κινουμένων σχεδίων sprite  Αφού αλλάξετε την ακολουθία, αν θέλετε να αλλάξει και η εικόνα, πρέπει να χρησιμοποιήσετε τον κωδικό 251	DB 254, 10, 0 Η ακολουθία 10 συνδέεται. Το μηδέν είναι ένα πληρωτικό Εάν η ακολουθία που έχει ήδη ανατεθεί στο sprite είναι αυτή που έχει ήδη, τότε δεν υπάρχει πρόβλημα (το id του πλαισίου δεν μηδενίζεται). Σε περίπτωση που θέλετε να μηδενίσετε το frame id, η τρίτη παράμετρος πρέπει να είναι 1, για παράδειγμα: DB 254, 10, 1
253	Αλλαγή εικόνας 	DB 253 DW new_img Συνδέεται η εικόνα "new_img", η οποία πρέπει να είναι μια διεύθυνση μνήμης.
252	Αλλαγή διαδρομής	DB 252, 2, 0 Η διαδρομή 2 συνδέεται
251	Πηγαίνετε στο επόμενο πλαίσιο από το animation. 	DB 251, 0, 0 Το Sprite είναι κινούμενο. Τα δύο μηδενικά είναι fillers

ΣΗΜΑΝΤΙΚΟ: προσέξτε πολύ να γράφετε DB και DW εκεί που πρέπει να χρησιμοποιηθούν, δηλαδή, για παράδειγμα, αν αλλάξετε εικόνες, θα πρέπει να προηγείται της εικόνας το DW και όχι το DB. Εάν κάνετε ένα τέτοιο λάθος, η διαδρομή σας δεν θα λειτουργήσει.

ΣΗΜΑΝΤΙΚΟ: οι κωδικοί διαφυγής μπορούν να χρησιμοποιηθούν στη μέση μιας διαδρομής, αλλά το τελευταίο τμήμα δεν μπορεί να είναι κωδικός διαφυγής, πρέπει να είναι μια κίνηση, ακόμη και αν είναι ακίνητο, κάτι σαν "DB 1,0,0,0".

12.2.1 Αναγκαστικές αλλαγές κατάστασης από διαδρομές

Αυτή η δυνατότητα είναι πολύ ενδιαφέρουσα για την επιτάχυνση των παιχνιδιών σας και είναι διαθέσιμη από την V27. Σημαίνει ότι μπορούμε να επιβάλλουμε μια αλλαγή κατάστασης στη μέση μιας διαδρομής. Για να το κάνουμε αυτό, υποδεικνύουμε ότι θέλουμε μια αλλαγή κατάστασης υποδεικνύοντας τον αριθμό των βημάτων στο τμήμα ως τιμή = 255.

Η αλλαγή κατάστασης είναι ένα ακόμη τμήμα και είναι σημαντικό να διατηρήσετε τον ίδιο αριθμό παραμέτρων ανά τμήμα, δηλαδή 3 bytes. Μια αλλαγή κατάστασης σε status=13 θα μπορούσε να γραφτεί ως εξής:

255,13,0

Η τρίτη παράμετρος (το μηδέν) δεν έχει καμία σημασία, είναι απλώς ένα "γέμισμα" για να κάνει το τμήμα να μετράει 3 bytes, αλλά είναι απαραίτητο.

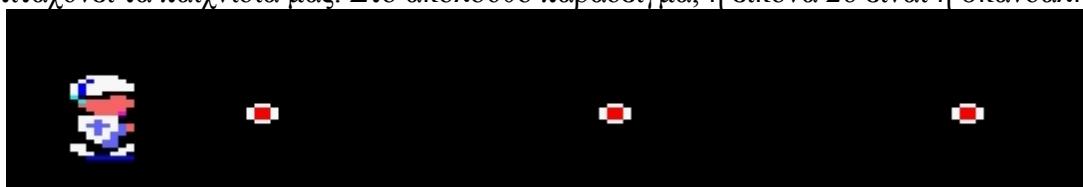
Η τιμή 255 θα πει στην εντολή **|ROUTEALL** ότι αυτό που πρέπει να κάνει αυτή τη φορά είναι να αλλάξει την κατάσταση του sprite, αναθέτοντάς του την κατάσταση που υποδεικνύεται παρακάτω. Η αλλαγή κατάστασης εκτελείται χωρίς να καταναλώνεται βήμα, οπότε το επόμενο βήμα μετά την αλλαγή κατάστασης θα εκτελείται πάντα. Αν δεν θέλουμε να κινείται πλέον το sprite, απλά ορίζουμε ένα τμήμα ενός βήματος χωρίς κίνηση στο X ή στο Y αμέσως μετά την αλλαγή κατάστασης. Ας δούμε ένα παράδειγμα:

```
ROUTE3; fire_dere
-----
db 40,0,2- σαράντα βήματα προς τα δεξιά με Vx=2
db 255,0,0- αλλαγή κατάστασης σε μηδέν
db 1,0,0- ακόμα Vy=0, Vx=0 db
0

ROUTE4; trigger_izq
-----
db 40,0,-2- σαράντα βήματα προς τα αριστερά με
Vx=-2 db 255,0,0- αλλαγή κατάστασης στο μηδέν
db 1,0,0- ακόμα Vy=0, Vx=0 db
0
```

Θα χρησιμοποιήσουμε αυτές τις δύο διαδρομές για να πυροβολήσουμε με τον χαρακτήρα μας. Η πρώτη, αφού περάσει από 40 βήματα στα οποία προχωράει 2 bytes στο X, υφίσταται μια αλλαγή κατάστασης και το sprite πηγαίνει στην κατάσταση 0, δηλαδή απενεργοποιημένο. Το επόμενο τμήμα έχει μόνο ένα βήμα και καμία κίνηση (vy=0, vx=0).

Με αυτόν τον μηχανισμό μπορούμε να ενεργοποιούμε και να απενεργοποιούμε τα εναύσματα όταν φεύγουν από την οθόνη. Αυτό εξοικονομεί τη λογική της BASIC και επιταχύνει τα παιχνίδια μας. Στο ακόλουθο παράδειγμα, η εικόνα 26 είναι η σκανδάλη.



Σχ. 71 Εναύσματα με αλλαγή κατάστασης κατά τη διαδρομή

```
10 MNHMH 24999
20 MODE 0: DEFINT A-Z: CALL &6B78:' install RSX
25 ON BREAK GOSUB 280
30 CALL &BC02:'επαναφορά της προεπιλεγμένης παλέτας για παν
ενδεχόμενο'.
40 INK 0,0: "μαύρο φόντο".
50 FOR j=0 TO 31:|SETUPSP,j,0,&X0:NEXT:'επαναφορά των sprites
80 |SETLIMITS,12,80,0,186: ' ορίστε τα όρια της οθόνης του
παιχνιδιού.
90 x=40:y=100:' συντεταγμένες του χαρακτήρα
100 |SETUPSP,0,0,1:' κατάσταση χαρακτήρα
110 |SETUPSP,0,7,1:dir=1:'Αικολουθίζει κινούμενων σχεδίων που
εκχωρείται κατά την έναρξη
120 |LOCATESP,0,y,x:'Τοποθετήστε το sprite (χωρίς να το εκτυπώσετε
ακόμα)
```

```

125 |MUSIC,0,0,6
126 for i=1 έως 4:|SETUPSP,10+i,9,26:next:'shots
130 Ο κύκλος του παιχνιδιού...
150 |AUTOALL,1:|PRINTSPALL,0,0
170 ' character movement routine -----
    AN INKEY(27)=0 TOTE AN dir=2 TOTE dir=1:|SETUPSP,0,7,dir
    ELSE
    |ANIMA,0:x=x+1:GOTO 191
190 EAN INKEY(34)=0 TOTE EAN dir=1 TOTE dir=2:|SETUPSP,0,7,dir
ELSE
|ANIMA,0:x=x-1
191 IF wait<cycle-10 then if INKEY(47)=0 THEN wait=cycle:disp= 1+ disp
mod 4 :|LOCATESP,10+disp,y+8,x: |SETUPSP,10+disp,0,137:
|SETUPSP,10+disp,15,2+dir
200 |LOCATESP,0,y,x
201 κύκλος=κύκλος+1
210 goto 150
280 |MUSIC:MODE 1: INK 0,0:PEN 1

```

Οι αλλαγές κατάστασης μπορούν να εξαναγκαστούν σε οποιοδήποτε τμήμα της διαδρομής, όχι απαραίτητα στο τέλος, αν και στην περίπτωση ενός σκανδάλου είναι πολύ λογικό να γίνει αυτό στο τέλος της διαδρομής.

12.2.2 Αναγκαστικές αλλαγές ακολουθίας από διαδρομές

Μπορούμε να αλλάξουμε την ακολουθία κίνησης ενός sprite χρησιμοποιώντας ένα ειδικό τμήμα. Όταν βάλουμε 254 στην τιμή του αριθμού των βημάτων, η εντολή ROUTEALL θα ερμηνεύσει ότι πρέπει να γίνει αλλαγή της ακολουθίας κινουμένων σχεδίων στο sprite. Παράδειγμα:

254,10,0

Αυτό το τμήμα αλλάζει την ακολουθία κίνησης του sprite, θέτοντας τον αριθμό ακολουθίας 10. Η τρίτη παράμετρος (το μηδέν) σημαίνει ότι η ακολουθία δεν θα επανεκκινηθεί, οπότε αν το sprite βρίσκεται στο καρέ 5 μιας άλλης ακολουθίας, θα ανατεθεί η νέα ακολουθία και το καρέ της ντίσκο θα διατηρηθεί, αν και τώρα θα αντιστοιχεί σε άλλη εικόνα. Αυτό είναι πολύ χρήσιμο γιατί μπορούμε να αναθέσουμε μια ακολουθία σε ένα sprite μέσα σε μια διαδρομή και αν την είχε ήδη, είναι ακίνδυνο. Για να επαναφέρετε την εκχωρημένη ακολουθία (στο καρέ 0) πρέπει να βάλετε ένα 1 στην τρίτη παράμετρο:

254,10,1

Σε περίπτωση που αναθέτετε μια ακολουθία και θέλετε να την επανεκκινήσετε, καλό είναι να χρησιμοποιήσετε τον κώδικα κίνησης, το 251 που θα δούμε αργότερα. Με αυτόν τον τρόπο, η εικόνα που σχετίζεται με το sprite θα αλλάζει στον πίνακα sprite και όχι μόνο το **frame_id**.

Όπως και με την αλλαγή κατάστασης, η αλλαγή ακολουθίας εκτελείται χωρίς να καταναλώνεται βήμα, οπότε το επόμενο βήμα της αλλαγής ακολουθίας θα εκτελείται πάντα.

12.2.3 Αναγκαστικές αλλαγές εικόνας από διαδρομές

Είδαμε πώς να δρομολογείτε sprites με το ROUTEALL ή ακόμα καλύτερα με το AUTOALL,¹

Συχνά δεν θέλουμε να δρομολογήσουμε ένα sprite μέσω μιας τροχιάς αλλά κάτι πιο καθημερινό: να πηδήξουμε με έναν χαρακτήρα. Στο παράδειγμα του κεφαλαίου 9 είδαμε πώς να το κάνουμε αυτό με έναν πίνακα BASIC που περιέχει τις σχετικές κινήσεις της συντεταγμένης Y. Σε αυτή την περίπτωση θα κάνουμε το ίδιο με μια διαδρομή, επιτυγχάνοντας μια ταχύτερη κίνηση.



Εικ. 72 Παράλειψη με χρήση διαδρομής

Για να μην χρειάζεται να ελέγξουμε αν ο χαρακτήρας έχει φτάσει στο ζενίθ του άλματος, μπορούμε να χρησιμοποιήσουμε ένα ειδικό τμήμα που υποδεικνύει την αλλαγή εικόνας. Όπως κάθε άλλο τμήμα, καταναλώνει 3 bytes, αλλά σε αυτή την περίπτωση το πρώτο είναι ο δείκτης αλλαγής εικόνας (μια τιμή 253) και τα επόμενα δύο αντιστοιχούν στη διεύθυνση μνήμης της εικόνας. **ΠΡΟΣΟΧΗ**, πρέπει να χρησιμοποιήσετε το "**dw**" πριν από το όνομα της εικόνας που θέλετε να διαθέσετε, οπότε θα πρέπει να γράψετε αυτό το τμήμα αλλαγής εικόνας σε δύο γραμμές. Ένα "**db**" για το 253 και ένα "**dw**" για τη διεύθυνση μνήμης της εικόνας.

db 253
dw SOLDIER_R1_UP

Στο ακόλουθο παράδειγμα έχουμε ένα ομοίωμα που πηδάει. Στην πορεία προς τα πάνω το ομοίωμα διαγράφεται από κάτω, ενώ στην πορεία προς τα κάτω διαγράφεται από πάνω. Για να μην υπάρχει ασυνέχεια στην κίνηση, ακριβώς κατά την αλλαγή της μιας εικόνας με την άλλη είναι απαραίτητο να επαναπροσδιορίσουμε κάθετα τον στρατιώτη, ανεβάζοντας την εικόνα προς τα κάτω ακριβώς 5 γραμμές, ώστε να συμπέσει με τον στρατιώτη που ανέβαινε.

Αυτό θα ήταν το αρχείο sequences_mygame.asm

;=====

έως και 31 ακολουθίες κινουμένων σχεδίων

;=====

πρέπει να είναι σταθερός πίνακας και όχι μεταβλητός πίνακας
; κάθε ακολουθία περιέχει τις διευθύνσεις των κυκλικών πλαισίων
κινουμένων σχεδίων

Κάθε ακολουθία αποτελείται από 8 διευθύνσεις μνήμης εικόνας.

Ζυγός αριθμός επειδή οι κινήσεις είναι συνήθως ζυγός αριθμός
ένα μηδέν σημαίνει τέλος της ακολουθίας, αν και χρησιμοποιούνται πάντα 8
λέξεις.

Όταν βρεθεί μηδέν, γίνεται νέα αρχή.
εάν δεν υπάρχει μηδέν, μετά το πλαίσιο 8 αρχίζει πάλι.

Η μηδενική ακολουθία είναι ότι δεν υπάρχει ακολουθία.

Ξεκινάμε από την ακολουθία 1

;-----**animation sequences** -----
_SEQUENCES_LIST

```
dw SOLDIER_R0,SOLDIER_R2,SOLDIER_R1,SOLDIER_R2,0,0,0,0,0,0 ; 1
dw SOLD_L0,SOLD_L2,SOLD_L1,SOLD_L2,0,0,0,0,0,0 ; 2 dw
SOLD_R1_UP,0,0,0,0,0,0,0,0;3
dw
SOLDIER_R1_DOWN,0,0,0,0,0,0,0,0,0
,0,0,0;4 dw
SOLDIER_L1_UP,0,0,0,0,0,0,0,0,0;5
dw
SOLDIER_L1_DOWN,0,0,0,0,0,0,0,0,0
;6
```

_MACRO_SEQUENCES

;-----MACRO SEQUENCES -----

είναι ομάδες ακολουθιών, μία για κάθε κατεύθυνση.

; το νόημα είναι:

; ακόμα, αριστερά, δεξιά, πάνω, πάνω-αριστερά, πάνω-δεξιά, κάτω,

κάτω-αριστερά, κάτω-δεξιά

Οι αριθμοί αριθμούνται από το 32 και μετά

db 0,2,1,3,5,3,3,4,6,4; 32 --> soldier sequences , id=32. το επόμενο θα

ήταν το 33

Θα χρησιμοποιήσουμε δύο διαδρομές, μία για να πηδήξουμε δεξιά και μία για να πηδήξουμε αριστερά. Αυτό θα είναι το αρχείο routes_mygame.asm.

; ΛΙΣΤΑ ΔΙΑΔΡΟΜΩΝ

=====

**βάλτε εδώ τα ονόματα όλων των διαδρομών που
κάνετε ROUTE_LIST**

dw ROUTE0
dw ROUTE1
dw ROUTE2
dw ROUTE3
dw ROUTE4
dw ROUTE4

ΟΡΙΣΜΟΣ ΚΑΘΕ ΔΙΑΔΡΟΜΗΣ

=====

ROUTE0; jump_right

db 253
dw SOLDIER_R1_UP
db 1,-5,1
db 2,-4,1
db 2,-3,1
db 2,-2,1
db 2,-1,1
db 253
dw SOLDIER_R1_DOWN
db 1,-5,1; πάνω, ώστε UP και κάτω να
ταιριάζουν db 2,1,1
db 2,2,1
db 2,3,1
db 2,4,1
db 1,5,1
db 253
dw SOLDIER_R1
db 1,5,1; κατεβείτε ένα ακόμα, καθώς το R1 δεν έχει μαύρα στην κορυφή.
db 255,13,0- νέα κατάσταση, χωρίς σημαία διαδρομής και με σημαία
κίνησης db 254,32,0- ακολουθία μακροεντολών 32
db 1,0,0; quietooo!!!!
db 0

ROUTE1; jump_left

db 253
dw SOLDADO_L1_UP
db 1,-5,-1
db 2,-4,-1
db 2,-3,-1
db 2,-2,-1
db 2,-1,-1
db 253
dw SOLDIER_L1_DOWN
db 1,-5,-1- ανύψωση για να ταιριάζει προς τα
πάνω και προς τα κάτω db 2,1,-1

```

db 2,2,-1
db 2,3,-1
db 2,4,-1
db 1,5,-1
db 253
dw SOLDIER_L1
db 1,5,-1; ακόμα ένα κάτω
db 255,13,0- νέα κατάσταση, χωρίς σημαία διαδρομής και με σημαία
κίνησης db 254,32,0- ακολουθία μακροεντολών 32
db 1,0,0; stillooo.!!!!
db 0

ROUTE2; πουλί
db 30,0,-1
db 10,0,0
db 20,2,-1
db 20,-2,-1
db 0

ROUTE3; fire_dere
;-----
db 40,0,2
db 255.0
db 1,0,0
db 0

ROUTE4; trigger_izq
;-----
db 40,0,-2
db 255.0
db 1,0,0
db 0

```

Και αυτό θα ήταν το πρόγραμμα-παράδειγμα. Συγκρίνετε την εκτέλεσή του με εκείνη του κεφαλαίου 7 για να δείτε τη διαφορά στην ταχύτητα. Θα παρατηρήσετε μια

μείωση της ταχύτητας.

```

20 MODE O: DEFINT A-Z: CALL &6B78:' install RSX
25 ON BREAK GOSUB 2800
30 CALL &BC02:ink 0,0:'επαναφέρει την προεπιλεγμένη παλέτα
50 FOR j=0 TO 31:|SETUPSP,j,0,0,0:NEXT:'επαναφορά των sprites
80 |SETLIMITS,12,80,0,186: ' ορίστε τα όρια της οθόνης του
παιχνιδιού.
90 x=40:y=100:jump=0:cycle=40:' συντεταγμένες χαρακτήρα
100 |SETUPSP,0,0,13:|SETUPSP,0,5,0,0:' κατάσταση χαρακτήρα
110 |SETUPSP,0,7,1:|SETUPSP,0,7,32:'Ακολουθία κίνησης που εκχωρείται
κατά την έναρξη

```

LOCATESP,0,y,x:"Τοποθετήστε το sprite (χωρίς	να το εκτυπώσετε ακόμα)
123 Locate 1,1: print "press Q" : print "stop" : print "for με διαδρομή"	skip": print "παράδειγμα
124 print "press SPACE to fire" print "press SPACE to fire" print "press SPACE to fire" print "press SPACE to fire" print "press SPACE to fire" print "press SPACE to fire" print	
125 PLOT 1,150:DRAW 640,150: PLOT 92,150:DRAW	92,400: "δάπεδο και

τοίχος

126 for i=1 έως 4:|SETUPSP,10+i,9,26:next:'shots
|MUSIC,0,0,5: 'Η μουσική αρχίζει να παιζει
130 ----- "κύκλος παιχνιδιού".
150 |AUTOALL,1:|PRINTSPALL,0,1,0
170 ' character movement routine -----

```

172 AN INKEY(47)=0 TOTE αν wait<cycle-10 τότε wait=cycle:disp= 1+
  disp mod
  4: |LOCATESP,10+disp,peek(27001)+8,peek(27003):|SETUPSP,10+disp,0,137:|
  SETUPSP,10+disp,15,3+dir
173 if peek(27000)>128 then 193 else |SETUPSP,0,6,0: ' if state is
  >128 είναι ότι πηδάω (έχει διαδρομή)
174 EAN INKEY(67)=0 THEN |SETUPSP,0,0,137:|SETUPSP,0,15,dir:'skip
180 IF INKEY(27)=0 THEN dir=0:|SETUPSP,0,6,1:'ir derecha
190 AN INKEY(34)=0 TOTE dir=1:|SETUPSP,0,6,-1:'ir αριστερά
193 κύκλος=κύκλος+1
310 goto 150
2800 |MUSIC:MODE 1: INK 0,0:PEN 1

```

Για να ελέγξω ότι το ομοίωμα πηδάει, απλά ρωτάω την κατάσταση με peek (27000). Με αυτόν τον τρόπο θα ξέρουμε αν έχει ενεργή τη σημαία δρομολόγησης και αν ναι, δεν θα περάσουμε από τις γραμμές που το μετακινούν αριστερά και δεξιά.

12.2.4 Αναγκαστική αναδρομολόγηση από διαδρομές

Μπορούμε να αλλάξουμε τη διαδρομή ενός sprite χρησιμοποιώντας ένα ειδικό τμήμα. Όταν βάλουμε 252 στην τιμή του αριθμού των βημάτων, η εντολή ROUTEALL θα ερμηνεύσει ότι πρέπει να γίνει αλλαγή διαδρομής στο sprite. Παράδειγμα:

252,2,0

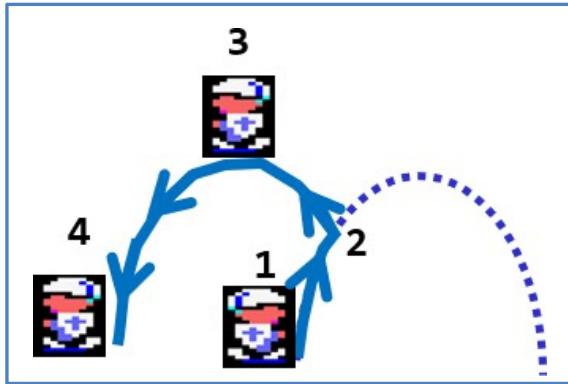
Αυτό το τμήμα αλλάζει τη διαδρομή του sprite, θέτοντας τον αριθμό διαδρομής 2. Η τρίτη παράμετρος (το μηδέν) δεν σημαίνει τίποτα, είναι απλώς ένα "γέμισμα" για να κάνει το τμήμα να μετράει 3 bytes, αλλά είναι απαραίτητο.

Όπως και με την αλλαγή κατάστασης, η αλλαγή διαδρομής εκτελείται χωρίς να καταναλώνεται βήμα, οπότε θα εκτελείται πάντα το επόμενο βήμα μετά την αλλαγή διαδρομής, το οποίο θα είναι το πρώτο βήμα του πρώτου τμήματος της νέας διαδρομής.

Δεδομένου ότι μια διαδρομή μπορεί να έχει μήκος το πολύ 255 bytes και ένα τμήμα έχει μήκος 3 bytes, μια διαδρομή μπορεί να έχει μήκος το πολύ 84 τμήματα. Μπορεί να χρειαστεί να δημιουργήσετε μια ακόμα μεγαλύτερη διαδρομή, και σε αυτή την περίπτωση μπορείτε να το κάνετε με τη συνένωση του τέλους μιας διαδρομής με μια αλλαγή διαδρομής σε μια άλλη διαδρομή, και μπορείτε να συνθέσετε όσες διαδρομές επιθυμείτε.

12.2.5 Αναγκαστικές αλλαγές διαδρομής από τη BASIC

Ας υποθέσουμε ότι θέλετε ο χαρακτήρας σας να πηδήξει προς τα δεξιά και στη μέση του άλματος θέλετε ο χαρακτήρας να μπορεί να αλλάξει προς τα αριστερά, συνεχίζοντας το άλμα. Αυτό που προτείνουμε μπορεί να αναπαρασταθεί με αυτό το σχέδιο:



Εικ. 73 Αλλαγή διαδρομής κατά τη διάρκεια ενός άλματος

Σε αυτό το παράδειγμα ο χαρακτήρας μας κάνει άλμα και στη μέση της διαδρομής του άλματος προς τα δεξιά, αλλάζει κατεύθυνση στο σημείο 2, συνεχίζοντας τη διαδρομή του άλματος προς τα αριστερά, αλλά χωρίς να ξεκινήσει νέο άλμα, απλά συνεχίζοντας το άλμα και φτάνοντας στο ίδιο ύψος (σημείο 3) που θα έφτανε με το άλμα προς τα δεξιά για να ολοκληρώσει τελικά το άλμα προς τα αριστερά στο σημείο 4.

Για να το κάνουμε αυτό πρέπει να αλλάξουμε το μονοπάτι του χαρακτήρα μας από τη BASIC στο σημείο 2, αλλά δεν μπορούμε να χρησιμοποιήσουμε την εντολή **|SETUPSP** γιατί αυτό θα αρχικοποιούσε το μονοπάτι, με αποτέλεσμα ένα πολύ μεγαλύτερο άλμα, ξεκινώντας από το σημείο 2. Αυτό που μπορούμε να κάνουμε σε αυτή την περίπτωση είναι απλά να **κάνουμε POKE** στη διεύθυνση μνήμης που αποθηκεύει το μονοπάτι του Sprite, η οποία είναι η διεύθυνση της κατάστασής του +15, όπως φαίνεται στον πίνακα χαρακτηριστικών sprite. Αυτό το POKE αλλάζει τη διαδρομή, αλλά διατηρεί τη θέση άθικτη (αριθμός τμήματος και θέση όπου βρίσκεται ο χαρακτήρας). ΠΡΟΣΟΧΗ: τα δύο μονοπάτια πρέπει να έχουν τα ίδια τμήματα και το ίδιο μήκος, διότι αν μεταβείτε σε ένα μικρότερο μονοπάτι και βρεθείτε σε ένα τμήμα που δεν υπάρχει σε αυτό το μονοπάτι, μπορεί να προκύψει ένα απρόβλεπτο αποτέλεσμα.

Υποθέτοντας ότι έχουμε δύο μονοπάτια άλματος (μονοπάτι 0 άλμα δεξιά και μονοπάτι 1 άλμα αριστερά), το ακόλουθο παράδειγμα απεικονίζει την έννοια:

```

130 ----- "κύκλος παιχνιδιού".
150 cycle=cycle+1: AUTOALL,1: |PRINTSPALL,0,1,0
170 ' character movement routine -----
173 ΕΑΝ PEEK(27000)<128 ΤΟΤΕ 178
174 βρισκόμαστε στη μέση ενός άλματος
175 IF INKEY(27)=0 THEN POKE 27015,0:GOTO 180:'αλλαγή διαδρομής προς τα δεξιά
    IF INKEY(34)=0 THEN POKE 27015,1:GOTO 180:'αλλαγή διαδρομής προς τα αριστερά
177 GOTO 150
178 IF INKEY(67)=0 THEN |SETUPSP,0,0,137:|SETUPSP,0,15,dir:'skip
    IF INKEY(27)=0 THEN dir=0:|SETUPSP,0,6,1:'ir derecha
190 ΑΝ INKEY(34)=0 ΤΟΤΕ dir=1:|SETUPSP,0,6,-1:'ir αριστερά
310 GOTO 150

```

Ο μόνος περιορισμός της χρήσης του POKE για το σκοπό αυτό είναι ότι η εικόνα του χαρακτήρα δεν αλλάζει μέχρι να συναντήσει έναν κωδικό αλλαγής εικόνας στη μέση της διαδρομής. Η αλλαγή της εικόνας με τη χρήση **|SETUPSP** είναι δυνατή αλλά επικίνδυνη, καθώς δεν γνωρίζετε αν ο χαρακτήρας πηγαίνει προς τα πάνω (σβήνει με τις κάτω γραμμές) ή προς τα κάτω (σβήνει με τις πάνω γραμμές). Επομένως, είναι προτιμότερο να αναθέσετε απλώς τη διαδρομή και να αφήσετε την ίδια τη διαδρομή να

αλλάξει την εικόνα το συντομότερο δυνατό. Μπορείτε ακόμη και να τοποθετήσετε στη μέση της διαδρομής αλλαγές εικόνας, ακόμη και αν δεν είναι απαραίτητες σε περίπτωση που εμφανιστεί αυτή η περίσταση. Αυτό που πρέπει να βεβαιωθείτε είναι ότι η εικόνα άλματος έχει bytes διαγραφής και στις δύο πλευρές, γιατί αν δεν έχει και αλλάξει κατεύθυνση, θα αφήσει ίχνη.

12.2.6 Αναγκαστική κίνηση από διαδρομές

Μπορούμε να αφήσουμε τη σημαία κίνησης ενός Sprite ανενεργή και να το κινήσουμε μόνο σε ορισμένες στιγμές της διαδρομής χρησιμοποιώντας τον κώδικα 251:

251,0,0

Αυτό μπορεί να είναι πολύ χρήσιμο για να δώσετε την αίσθηση ενός sprite που πλησιάζει σε παιχνίδια που χρησιμοποιούν τεχνικές ψευδο-3D. Για παράδειγμα, για έναν μετεωρίτη που πλησιάζει και θέλουμε να αλλάξουμε τα καρέ για να φαίνεται μεγαλύτερος. Ο μετεωρίτης θα κινηθεί μερικές φορές πριν μετακινηθεί στο επόμενο μέγεθος. Αυτός ο μηχανισμός είναι πολύ παρόμοιος με τον μηχανισμό αλλαγής εικόνας, με τη διαφορά ότι σας επιτρέπει να ορίσετε την αλλαγή εικόνας χωρίς να προσδιορίσετε ρητά την εικόνα, αλλά απλά υποδεικνύοντας μια αλλαγή καρέ στην ακολουθία



κινούμενων σχεδίων που έχει ανατεθεί στο sprite.

Εικ. 74 Αναγκαστική κίνηση από τη διαδρομή

Με αυτή τη σημαία μπορείτε να χρησιμοποιήσετε την ίδια διαδρομή για ένα διαστημικό πουλί που πλησιάζει ή έναν μετεωρίτη. Με τη μη αναγραφή της εικόνας, σε κάθε περίπτωση θα εφαρμοστεί η εικόνα που αντιστοιχεί στην ακολουθία που έχει το κάθε sprite.

12.2.7 Πώς να δημιουργήσετε "δυναμικές" (όχι προκαθορισμένες) διαδρομές

Μια δυναμική διαδρομή είναι μια διαδρομή της οποίας η διαδρομή αποφασίζεται κατά την εκτέλεση του προγράμματος BASIC. Είναι χρήσιμη όταν το πρόγραμμά σας δημιουργεί δυναμικά λαβύρινθους ή πίστες αγώνων από τις οποίες πρέπει να περάσει ένας εχθρός και οι οποίες δεν είναι γνωστές εκ των προτέρων και επομένως δεν μπορούν να οριστούν στο αρχείο "routes_mygame.asm".

Για να φτιάξουμε μια τέτοια διαδρομή και να την αναθέσουμε σε ένα Sprite, αυτό που πρέπει να κάνουμε είναι να δημιουργήσουμε μια "κενή" διαδρομή στο αρχείο "routes_mygame.asm", στην προκειμένη περίπτωση τη διαδρομή 2.

; ΛΙΣΤΑ ΔΙΑΔΡΟΜΩΝ

;=====

**Βάλτε εδώ τα ονόματα όλων των διαδρομών που
κάνετε ROUTE_LIST**

dw ROUTE0

dw ROUTE1

dw ROUTE2

dw ROUTE3

dw ROUTE4

dw ROUTE4

ΟΡΙΣΜΟΣ ΚΑΘΕ ΔΙΑΔΡΟΜΗΣ

;=====

ROUTE0; δεξιά αριστερά db

10,0,1

db 10,0,-1

db 0

ROUTE1; πάνω κάτω

db 10,-1,0

db 10,1,0

db 0

**ROUTE2; δυναμική
δρομολόγηση ds**

100

Δημιουργήσαμε τη διαδρομή 2 με 100 bytes ελεύθερα για συμπλήρωση από τη BASIC. Ενδέχεται το μονοπάτι που θα δημιουργήσουμε να καταλαμβάνει λιγότερα και σε αυτή την περίπτωση αρκεί να κρατήσουμε λιγότερα bytes.

Αφού συναρμολογηθούν η βιβλιοθήκη και τα γραφικά, πρέπει να αναζητήσουμε τη διεύθυνση μνήμης της ετικέτας "ROUTE2" στο παράθυρο συμβόλων winape. Όταν την έχουμε, από τη BASIC θα προγραμματίσουμε τη διαδρομή χρησιμοποιώντας POKE από αυτή τη διεύθυνση μνήμης και τις ακόλουθες

Η POKE τοποθετεί ένα byte σε μια διεύθυνση μνήμης. Οι αριθμοί μας πρέπει να ανήκουν στην περιοχή -127..128 και η POKE δεν επιτρέπει την τοποθέτηση αρνητικών αριθμών. Για να το κάνετε αυτό πρέπει να χρησιμοποιήσετε τη θετική τιμή με την οποία η Amstrad αναπαριστά εσωτερικά τα αρνητικά (δηλαδή το συμπλήρωμα των δύο). Για να το κάνετε αυτό χρειάζεστε απλώς μια πράξη AND 255

10 A=-10

20 PRINT A: REM αυτό εκτυπώνει ένα 10

30 PRINT A AND 255: REM αυτό εκτυπώνει ένα 246 που είναι το ίδιο -10

Εν ολίγοις, αν θέλετε να εισαγάγετε ένα -10 πρέπει να εισαγάγετε ένα 246 και να χρησιμοποιήσετε την ίδια στρατηγική για κάθε αρνητικό αριθμό. Μην ξεχνάτε ότι το τελευταίο POKE της διαδρομής πρέπει να είναι η εισαγωγή ενός μηδενός, που σημαίνει τέλος της διαδρομής.

12.2.8 Προγραμματισμός διαδρομών, συμπεριλαμβανομένων των μοτίβων

Ας υποθέσουμε ότι θέλετε να φτιάξετε ένα μονοπάτι για έναν εχθρό που θα διασχίζει την οθόνη από τα δεξιά προς τα αριστερά ξανά και ξανά, υποθέτοντας ότι ξεκινάμε από μια αρχική θέση όπου το sprite βρίσκεται στο άκρο δεξιά της οθόνης.

ROUTE0; απλή διαδρομή

DB 80,0,-1 ; 80 βήματα. Σε κάθε βήμα μετακινείται 1 byte

DB 1,0,80 ; επανατοποθετήστε το sprite στην αρχική του

Θέση DB 0

Αυτή η διαδρομή μετακινεί ένα sprite με βήμα 1 byte σε κάθε καρέ. Αν θέλαμε να το επιβραδύνουμε, μετακινώντας 1 byte κάθε δεύτερο καρέ, θα μπορούσαμε να κάνουμε τα εξής:

ROUTE0; όχι τόσο απλή

διαδρομή DB 1,0,-1

DB 1,0,0

DB 0

Τώρα αυτή η διαδρομή μας επιτρέπει να μετακινήσουμε ένα sprite πιο αργά, αλλά δεν μπορούμε να επαναποθετήσουμε το sprite στην αρχική του θέση. Αυτό συμβαίνει επειδή, καθώς η οθόνη έχει πλάτος 80 bytes, χρειαζόμαστε 160 τμήματα, καθώς το sprite προχωράει κατά 1 byte κάθε δύο τμήματα. Το μονοπάτι που θα προκύψει θα ήταν πολύ μεγάλο και στην πραγματικότητα θα έπρεπε να συνδέσουμε 2 μονοπάτια, επειδή ένα μονοπάτι μπορεί να έχει μέγεθος μόνο 255 bytes (84 τμήματα).
Η βέλτιστη λύση είναι να ορίσετε μια σύντομη διαδρομή χωρίς να επαναποθετήσετε το sprite και να το επαναποθετήσετε από τη BASIC, χρησιμοποιώντας κάτι σαν αυτό:

```
80 |SETUPSP,31, 0, 128+16+1: REM περιστρεφόμενη, αυτόματη μετακίνηση, εκτυπώσιμη
85 |LOCATESP,31,100,80: ' τοποθετείται στη δεξιά πλευρά της οθόνης.
Κύκλος παιχνιδιού 90 rem
100 κύκλος=κύκλος +1
110 |AUTOALL,1: |PRINTSPALL
120 if MOD cycle 160=0 THEN |LOCATESP,31,100,80: ' επαναποθέτηση
130 goto 100
```

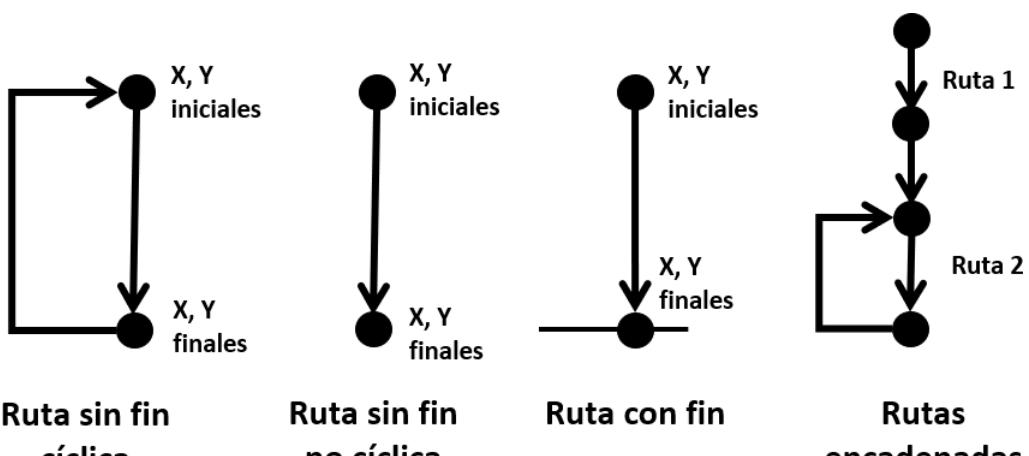
Αυτού του είδους οι στρατηγικές είναι χρήσιμες κάθε φορά που δεν θέλουμε να επαναλαμβάνουμε την ίδια κίνηση σε κάθε καρέ, αλλά θέλουμε να ορίσουμε ένα επαναλαμβανόμενο μοτίβο στο οποίο σε ορισμένα καρέ το sprite κινείται προς μια κατεύθυνση και σε άλλα κινείται προς μια άλλη κατεύθυνση ή και καθόλου. Ας δούμε ένα άλλο παράδειγμα, ένα κεκλιμένο μονοπάτι στο οποίο για κάθε 3 κατακόρυφες κινήσεις κάνουμε μία οριζόντια κίνηση.

ROUTE0; κεκλιμένη
διαδρομή DB 2,1,0
DB 1,1,1,1,1
DB 0

12.2.9 Τυπολογία των διαδρομών

Με όλα αυτά που είδαμε μπορούμε να ταξινομήσουμε τις διαδρομές στους ακόλουθους τύπους:

- **Κυκλικές ατελείωτες διαδρομές:** επαναποθετούν το sprite ή απλά καταλήγουν στις ίδιες συντεταγμένες από όπου ξεκίνησαν.
- **Μη κυκλικά ατελείωτα μονοπάτια:** προχωρούν επ' άπειρον και δεν επαναποθετούν το sprite, οπότε μπορούν να απομακρυνθούν άπειρα από την περιοχή παιχνιδιού, εκτός αν επαναποθετήσουμε το sprite από τη BASIC.
- **Δρομολογήσεις με τέλος:** στο τελευταίο βήμα αλλάξτε την κατάσταση του sprite απενεργοποιώντας τη σημαία δρομολόγησης.
- **Αλυσιδωτές διαδρομές:** από μια διαδρομή μπορείτε να μεταπηδήσετε σε μια άλλη διαδρομή και αυτή η δεύτερη διαδρομή μπορεί να είναι κυκλική ή μη κυκλική ή να έχει ένα τέλος ή ακόμη και να καταλήγει σε μια τρίτη διαδρομή.

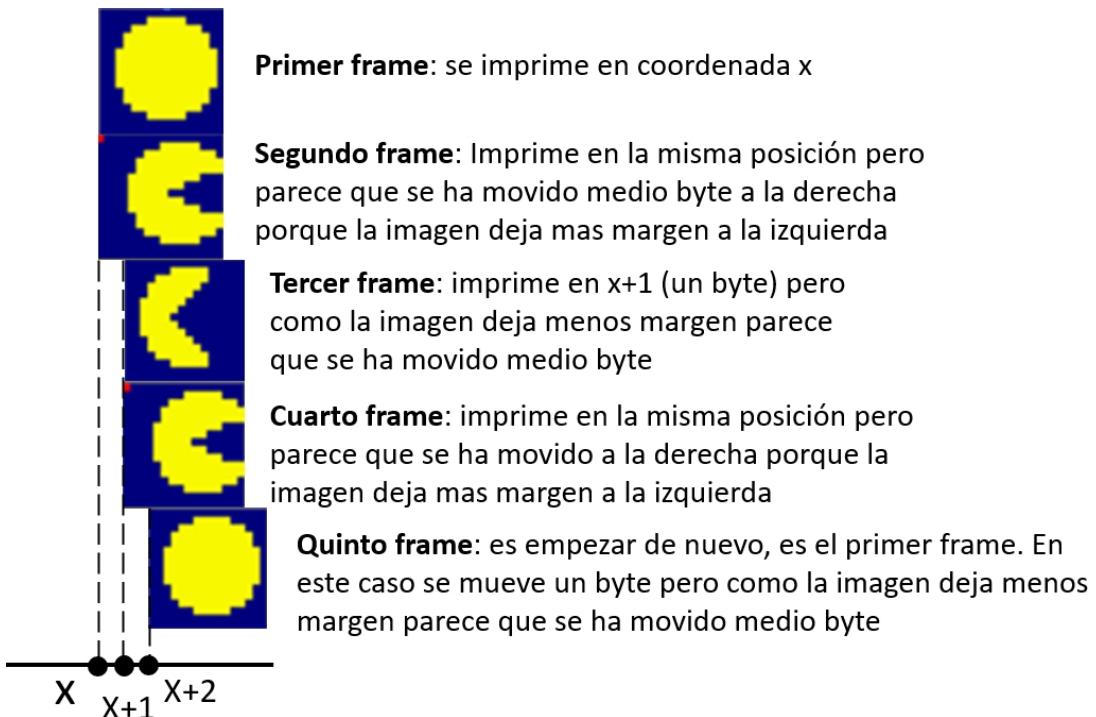


Σχ. 75 Τύποι διαδρομών

13 Ομαλή κίνηση μισού byte

Το 8BP μετακινεί τα sprites ανά byte με εντολές όπως η MOVE, και το σύστημα συντεταγμένων του είναι bytes, όχι pixels. Επομένως, μιλάμε για 80 θέσεις στον οριζόντιο άξονα και 200 στον κατακόρυφο άξονα.

Ένα byte περιέχει 2 εικονοστοιχεία στη λειτουργία 0 ή 4 εικονοστοιχεία στη λειτουργία 1. Μπορεί να θέλουμε μια πιο ομαλή κίνηση (εικονοστοιχείο προς εικονοστοιχείο στη λειτουργία μηδέν ή δύο εικονοστοιχεία στη λειτουργία 1). Υπάρχει ένα απλό τέχνασμα που μπορούμε να χρησιμοποιήσουμε γι' αυτό. Είναι να έχουμε μια εικόνα του χαρακτήρα μετατοπισμένη κατά μισό byte και απλά να την αναθέσουμε στο Sprite. Ακόμη και αν εκτυπωθεί στις ίδιες συντεταγμένες, θα φαίνεται σαν να έχει μετακινηθεί.

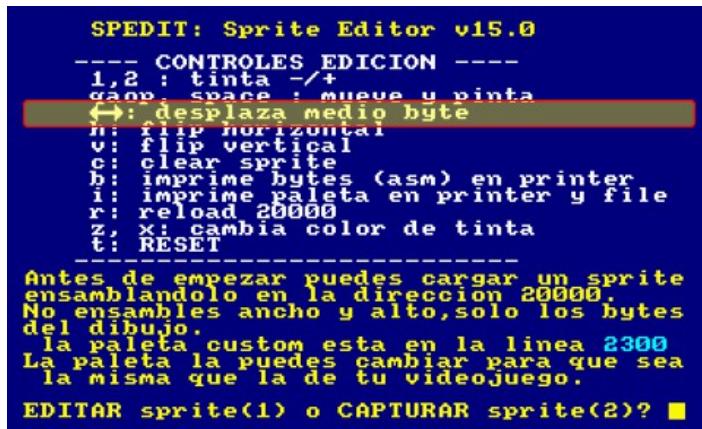


Σε αυτή την ακολουθία κινούμενων σχεδίων, υπάρχουν στιγμές που ο Pac-Man δεν κινείται, αλλά καθώς αλλάζουμε την εικόνα, είναι σαν να κινείται 2 pixel (μισό byte). Τις στιγμές που κινείται κατά 1 byte, η εικόνα εμφανίζεται μετατοπισμένη προς τα αριστερά κατά μισό byte, οπότε το "καθαρό" αποτέλεσμα είναι επίσης σαν να κινείται κατά 2 pixels (μισό byte). Για να ορίσουμε την κίνηση αυτού του Pac-Man μπορούμε να χρησιμοποιήσουμε τον μηχανισμό διαδρομής 8BP. Αυτό θα ήταν το παράδειγμα της διαδρομής της κίνησης προς τα δεξιά:

```
ROUTE0; db
```

```
    253  
    dw COCO_R1  
    db 1,0,0,0  
    db 253  
    dw COCO_R2  
    db 1,0,1  
    db 253  
    dw COCO_R1  
    db 1,0,0,0  
    db 253  
    dw COCO_R0  
    db 1,0,1  
    db 0
```

Η σχεδίαση εκτοπισμένων εικόνων μπορεί να είναι λίγο κουραστική. Γι' αυτό από την έκδοση **V15 του SPEDIT** έχετε ένα μηχανισμό για να μετατοπίζετε μια εικόνα κατά μισό byte (2 pixel στη λειτουργία 1 ή ένα pixel στη λειτουργία 0) προς τα δεξιά ή προς τα αριστερά. Όπως μπορείτε να δείτε στο μενού SPEDIT εμφανίζεται μια νέα επιλογή: με τα βέλη του δρομέα μπορείτε να μετατοπίσετε την εικόνα που έχετε σχεδιάσει.



```
SPEDIT: Sprite Editor v15.0
---- CONTROLES EDICION ----
1,2 : tinta -/+ 
gano, space : mueve u pinta
→: desplaza medio byte
n: flip horizontal
v: flip vertical
c: clear sprite
b: imprime bytes (asm) en printer
i: imprime paleta en printer y file
r: reload 20000
z, x: cambia color de tinta
t: RESET
-----
Antes de empezar puedes cargar un sprite
ensamblandolo en la direccion 20000.
No ensambles ancho y alto, solo los bytes
del dibujo.
La paleta custom esta en la linea 2300
La paleta la puedes cambiar para que sea
la misma que la de tu videojuego.
EDITAR sprite(1) o CAPTURAR sprite(2)? ■
```

Αυτό μας επιτρέπει να μετακινήσουμε εύκολα μια εικόνα για να εφαρμόσουμε την περιγραφόμενη τεχνική της ομαλής κίνησης.

14 Παιχνίδια κύλισης

Η βιβλιοθήκη 8BP σας επιτρέπει να κάνετε κύλιση με διάφορους τρόπους που μπορούν να συνδυαστούν ταυτόχρονα, αν και η πιο σημαντική μέθοδος βασίζεται στην εντολή **|MAP2SP**. Οι διαθέσιμες τεχνικές συνοψίζονται παρακάτω:

- **Μέσω εντολών κίνησης μπλοκ sprite:** Ένας απλός τρόπος για κύλιση με την 8BP είναι να δημιουργήσουμε απλά διακοσμητικά sprites που θα ορίσουμε την κατάστασή τους ώστε να μετακινούνται με τις εντολές **|MOVERALL** ή/και **|AUTOALL**.
- **Χρήση του |MAP2SP:** Η ιδέα πίσω από την πολυκατευθυντική κύλιση που παρέχει το **|MAP2SP** στο 8BP είναι απλή: όλα τα στοιχεία που αναπαρίστανται στην οθόνη είναι sprites, οπότε τα στοιχεία του κόσμου που θα εκτυπώσουμε και θα μετακινήσουμε στην οθόνη είναι sprites των οποίων οι σχετικές εικόνες θα είναι βουνά, σπίτια, δέντρα ή οτιδήποτε άλλο χρειάζεστε για να φτιάξετε τον "κόσμο" σας. Για να επιλέξετε ένα τμήμα του κόσμου και να το μετατρέψετε σε μια λίστα sprites, χρησιμοποιείται η συνάρτηση **|MAP2SP**. Η συνάρτηση **|UMAP** σας επιτρέπει να ενημερώσετε τον κόσμο με ένα τμήμα ενός μεγαλύτερου κόσμου.
- **Χρησιμοποιώντας την εντολή |STARS, η οποία θα σας επιτρέψει να κάνετε πολυκατευθυντική κύλιση μιας τράπεζας 40 εικονοστοιχείων που μπορείτε να τοποθετήσετε όπου θέλετε και την οποία μπορείτε να μετακινήσετε σε διαφορετικά επίπεδα και με διαφορετικές ταχύτητες.**
- **Χρησιμοποιώντας την εντολή |RINK, η οποία θα σας επιτρέψει να περιστρέψετε ένα μοτίβο μελάνης, δίνοντας μια αίσθηση κίνησης προς τα εμπρός, την οποία μπορείτε να χρησιμοποιήσετε σε ορισμένους τύπους κύλισης, όπως η κίνηση ενός δαπέδου από τούβλα, νερό κ.λπ.**

14.1 STARS: κύλινδρος από αστέρια ή στικτή γη

Στη βιβλιοθήκη 8BP έχετε μια πολύ εύκολη στη χρήση λειτουργία για να δημιουργήσετε ένα εφέ φόντου με κινούμενα αστέρια, δίνοντας την αίσθηση κύλισης. Αυτή είναι η συνάρτηση **|STARS**. Αυτή η λειτουργία είναι σε θέση να μετακινήσει έως και 40 αστέρια ταυτόχρονα χωρίς να μεταβάλει τα sprites σας, έτσι ώστε να είναι σαν να περνούν "από κάτω".

|STARS,<αρχικό αστεριών>,<χρώμα>,<dy>,<dx>,<dx>. **αστέρι>,<αριθμός**

Έχετε μια τράπεζα αστεριών και μπορείτε να συνδυάσετε διάφορες εντολές STARS για να εργαστείτε με ομάδες αστεριών σε διαφορετικές ταχύτητες, δίνοντας την αίσθηση αεροπλάνων με διαφορετικό βάθος.

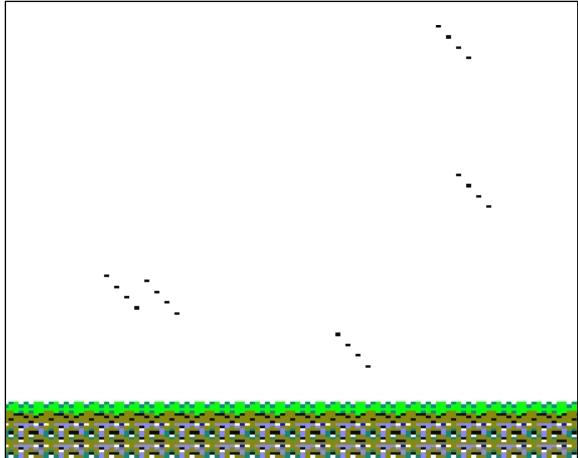
Η τράπεζα αστέρων αποτελείται από 40 ζεύγη bytes που αντιπροσωπεύουν τις συντεταγμένες (y,x). Καταλαμβάνουν τη διεύθυνση 42540 έως 42619 (80 bytes

συνολικά). Ένας τρόπος για να δημιουργήσουμε 40 τυχαία αστέρια θα ήταν (σημειώστε ότι αν έχουμε ήδη εκτελέσει το DEFINT A-Z ο αριθμός 42540 πρέπει να τεθεί σε δεκαεξαδικό σύστημα επειδή είναι μεγαλύτερος από το 32768).

```
FOR dir=42540 TO 42618 BHMA 2: POKE dir,RND*200: POKE  
dir+1,RND*80:NEXT
```

Για μια λεπτομερή περιγραφή του εντολή, ανατρέξτε στο κεφάλαιο "οδηγός αναφοράς". Σε αυτό το κεφάλαιο θα βρείτε διάφορα παραδείγματα για την προσομοίωση αστεριών, γης, αστεριών με δύο επίπεδα βάθους, βροχής ή ακόμα και χιονιού. Με φαντασία, είναι πιθανώς δυνατό να προσομοιώσετε περισσότερα πράγματα με την ίδια λειτουργία. Για παράδειγμα, αν τοποθετήσετε τα αστέρια σε ακολουθίες των 2 ή τριών εικονοστοιχείων διαγώνια, αντί να τα κατανέμετε τυχαία, μπορείτε να επιτύχετε μια μετατόπιση κίνησης "τμήματος", η οποία θα μπορούσε να είναι ιδανική.

για την προσομοίωση της βροχής.



Εικ. 76 Επίδραση βροχής με STARS

Καθώς το παράδειγμα ενός διπλού επιπέδου αστεριών βρίσκεται στο κεφάλαιο αναφοράς της βιβλιοθήκης, εδώ θα δούμε ένα παράδειγμα ενός διαστημοπλοίου που πετάει πάνω από έναν διάστικτο γήινο πλανήτη, με κάθετη κύλιση.



Σχ. 77 Στιγματισμένο φαινόμενο εδάφους με STARS

Υπάρχει ένας τρόπος για τη βελτιστοποιημένη κλήση της εντολής STARS, ο οποίος συνίσταται στην απλή κλήση της την πρώτη φορά με παραμέτρους και τις επόμενες φορές χωρίς παραμέτρους. Η εντολή θα υποθέσει ότι οι τιμές των παραμέτρων είναι οι ίδιες με εκείνες της τελευταίας κλήσης με παραμέτρους και έτσι εξοικονομείται χρόνος που δαπανά ο διερμηνέας BASIC για την επεξεργασία των παραμέτρων, έως και 1,7ms.

10 MNHMH 24999

11 'Εβαλα τυχαία αστέρια

12 FOR dir=42540 TO 42618 BHMA 2: POKE dir,RND*200:POKE dir+1,RND*80:NEXT

20 MODE 0: DEFINT A-Z: CALL &6B78:' install RSX

25 call &bc02:'restore default palette just in case'.

26 μελάνι 0.13:'γκρι φόντο

30 FOR j=0 TO 31:|SETUPSP,j,0,&X0:NEXT:'επαναφορά των sprites

40 |SETLIMITS,12,80,0,186: "Ορια της οθόνης του παιχνιδιού".

41 ' Θα δημιουργήσουμε ένα πλοίο στο sprite 31

42 |SETUPSP,31,0,&1:' κατάσταση

43 ship = &a2f8: |SETUPSP,31,9,ship:' ανάθεση εικόνας στο sprite 31

44 x=40:y=150: ' συντεταγμένες πλοίου

49 ' ---- κύκλος παιχνιδιού-----

50 |STARS,0,20,5,1,0:' μαύρα αστέρια σε γκρίζο έδαφος

55 gosub 100:' κίνηση του πλοίου

60 |PRINTSPALL,0,0

70 goto 50

99 ' συνήθης μετακίνηση πλοίου -----

100 AN INKEY(27)=0 TOTE x=x+1:GOTO 120

110 AN INKEY(34)=0 TOTE x=x-1

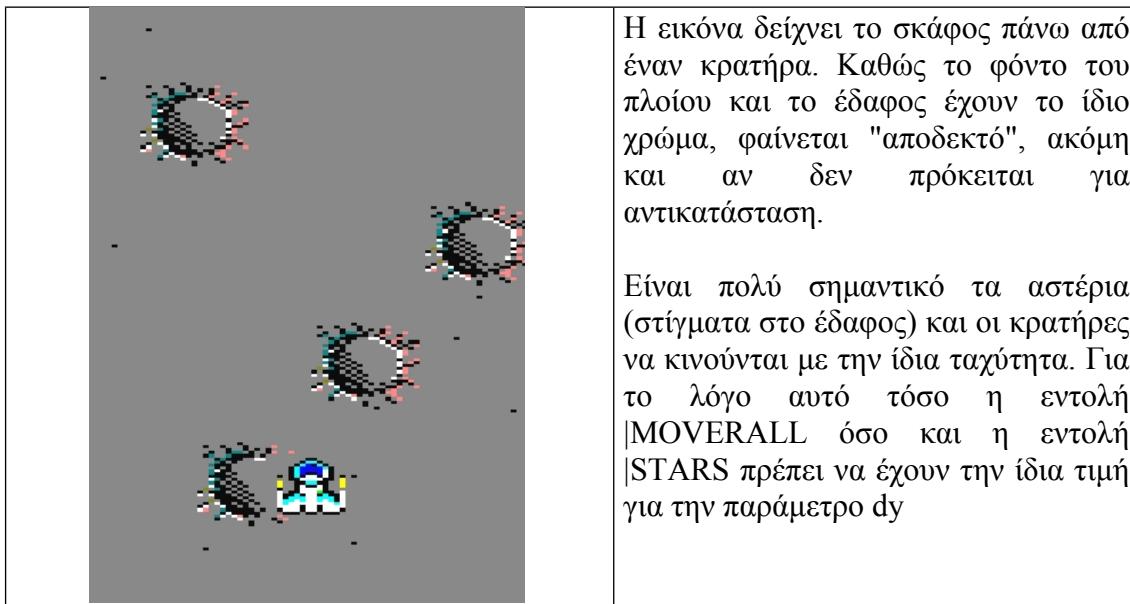
120 |LOCATESP,31,y,x

130 ΕΠΙΣΤΡΟΦΗ

14.2 Μετακινηθείτε χρησιμοποιώντας MOVERALL ή/και AUTOALL

Τώρα, κάνοντας συνδυασμένη χρήση της σχετικής κίνησης, της κύλισης με το αστέρι και της σειράς με την οποία εκτυπώνονται τα sprites, θα δούμε ένα παράδειγμα προσομοίωσης ενός διαστημοπλοίου που πετάει πάνω από ένα σεληνιακό τοπίο.

Πρώτα απ' όλα, επιλέξαμε το sprite 31 για το πλοίο μας, γιατί έτσι θα εκτυπωθεί τελευταίο. Τα sprites εκτυπώνονται με τη σειρά, ξεκινώντας από το μηδέν και καταλήγοντας στο 31. Αν ένας κρατήρας είναι sprite χαμηλότερα από το 31, θα εκτυπωθεί πριν από το πλοίο και το πλοίο θα είναι "πάνω" από αυτόν, δίνοντας την εντύπωση ότι πετάει πάνω από αυτόν.



Εικ. 78 Πετώντας πάνω από το φεγγάρι

Αυτός είναι ο κωδικός BASIC:

```
10 MNHMH 24999
11 'Εβαλα τυχαία αστέρια
12 FOR dir=42540 TO 42618 BHMA 2: POKE dir,RND*200:POKE
dir+1,RND*80:NEXT
20 MODE 0: DEFINT A-Z: CALL &6B78:' install RSX
25 call &bc02:'restore default palette just in case'.
26 μελάνι 0.13:'γκρι φόντο
30 FOR j=0 TO 31:|SETUPSP,j,0,&x0:NEXT:'επαναφορά των sprites
40 |SETLIMITS,12,80,0,186: ' όρια της οθόνης του παιχνιδιού

41 ' Θα δημιουργήσουμε ένα πλοίο στο sprite 31
42 |SETUPSP,31,0,&1:' κατάσταση
43 ship = &a2f8: |SETUPSP,31,9,ship:' ανάθεση εικόνας στο sprite 31
45 x=40:y=150: ' συντεταγμένες πλοίουν

46 ' τώρα οι κρατήρες
47 crater=&a39a: cy%=0
48 για i=0 έως 3 : |SETUPSP,i,9,crater:
49 |SETUPSP,i,0,&x10001: ' εντύπωση και σχετική κίνηση
50 x(i)=rnd*40+20:y(i)=i*40
60 |locatesp,i,y(i),x(i)
70 επόμενο
```

71 t=0

80 '----- κύκλος παιχνιδιού-----'

81 |STARS,0,20,5,3,0:' κίνηση μαύρων αστεριών

82 gosub 100:' κίνηση του πλοίου

83 |MOVERALL,3,0: "μετακίνηση κρατήρα".

84 t=t+1: if t > 10 then t=0:gosub 200:' έλεγχος κρατήρα

90 |PRINTSPALL,0,0:' εκτύπωση πλοίου και κρατήρα

91 επιλέξτε 81

99 ' συνήθης μετακίνηση πλοίου -----'

100 AN INKEY(27)=0 TOTE x=x+1:GOTO 120

110 AN INKEY(34)=0 TOTE x=x-1

120 |LOCATESP,31,y,x

130 ΕΠΙΣΤΡΟΦΗ

199 ' έλεγχος επανεισόδου σε κρατήρα

200 c=c+1: εάν c=6 τότε c=0

220 |PEEK,27001+c*16,@cy% 220 |PEEK,27001+c*16,@cy%

230 if cy%>200 then |POKE,27001+c*16,-20

240 επιστροφή

Ας δούμε ένα τελευταίο παράδειγμα που χρησιμοποιεί τη σχετική κίνηση για να δώσει την αίσθηση της κύλισης, χρησιμοποιώντας sprites με σχέδια σπιτιών, ένα στικτό πάτωμα και έναν χαρακτήρα που βρίσκεται στο κέντρο και, ανάλογα με την κατεύθυνση προς την οποία κινείται, κάνει τα πάντα να κινούνται γύρω του. Είναι ένα πολύ βασικό παράδειγμα, αλλά σας δίνει μια ιδέα για τις δυνατότητες αυτών των λειτουργιών - εδώ είναι ολόκληρη η πόλη που κινείται!



Eik. 79 Όλο το χωριό κινείται

10 MNHMH 24999

20 MODE 0: κλήση &6b78

30 DEFINT a-z

240 INK 0,12

241 σύνορα 7

250 FOR i=0 TO 31

260 |SETUPSP,i,0,&X0

270 ΕΠΟΜΕΝΟ

280 FOR i=0 TO 3

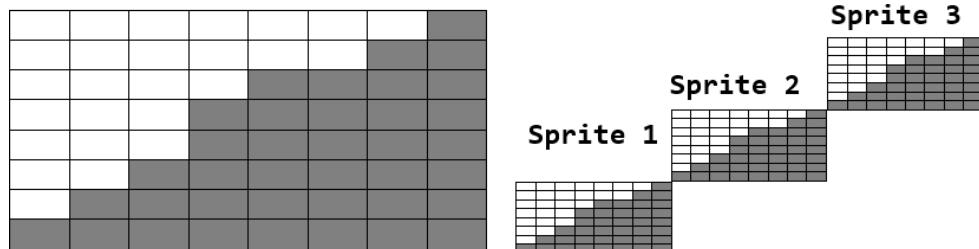
```

|SETUPSP,i,0,&X10001
|SETUPSP,i,9,&A01c:rem houses
301 |LOCATESP,i,RND*150+50,rnd*60+10
310 ΕΠΟΜΕΝΟ
320 |SETUPSP,31,7,6: χαρακτήρας rem
330 |LOCATESP,31,90,38
340 |SETUPSP,31,0,0,&X1111
xa=0:ya=0
410 AN INKEY(27)=0 TOTE xa=-1:
420 AN INKEY(34)=0 TOTE xa=+1:
430 AN INKEY(67)=0 TOTE ya=+2
440 AN INKEY(69)=0 TOTE ya=-2
450 |MOVERALL,ya,xa
460 |PRINTSPALL,1,0
470 |STARS,1,20,5,ήδη,xa
GOTO 410

```

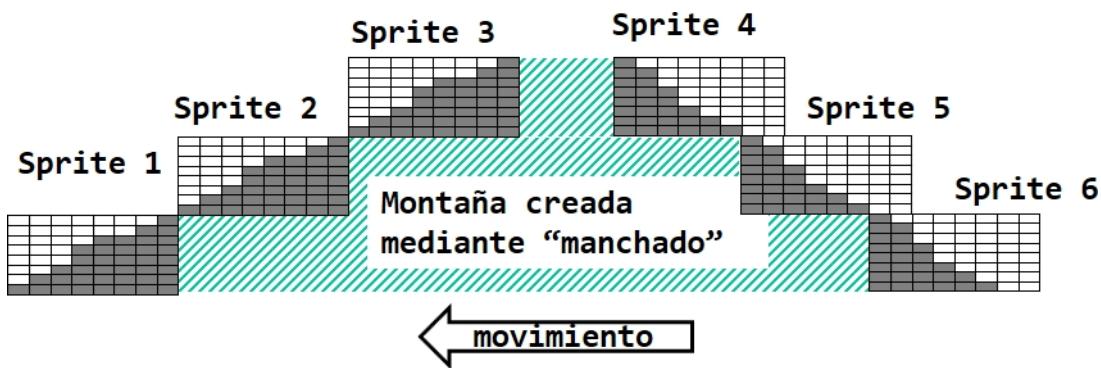
14.3 Τεχνική του "spotting"

Η τεχνική για τη ζωγραφική βουνών σε παιχνίδια με οριζόντια κύλιση και λιμνών σε παιχνίδια με κάθετη κύλιση είναι η ίδια. Αυτό που θα κάνουμε είναι να ζωγραφίσουμε μόνο την αρχή του βουνού, χρησιμοποιώντας ένα sprite για να ζωγραφίσουμε την αριστερή πλευρά του. Θα βάλουμε όσα περισσότερα θέλουμε. Σε αυτή την περίπτωση έχω βάλει τρία



Εικ. 80. Καθορισμός της κλίσης ενός βουνού με πολλά sprites

Κάνουμε το ίδιο με ένα είδωλο που θα συσχετίσουμε με 3 άλλα sprites και θα τα τοποθετήσουμε στα δεξιά, χτίζοντας τη δεξιά πλευρά του βουνού. Βεβαιωθείτε ότι το είδωλο έχει τουλάχιστον τις δύο τελευταίες στήλες των pixels στο μηδέν. Αυτό θα του επιτρέψει να σβήσει τον εαυτό του καθώς κινείται προς τα αριστερά. Σημειώστε ότι στην 8BP τα sprites κινούνται ανά byte (δύο pixel κάθε φορά).

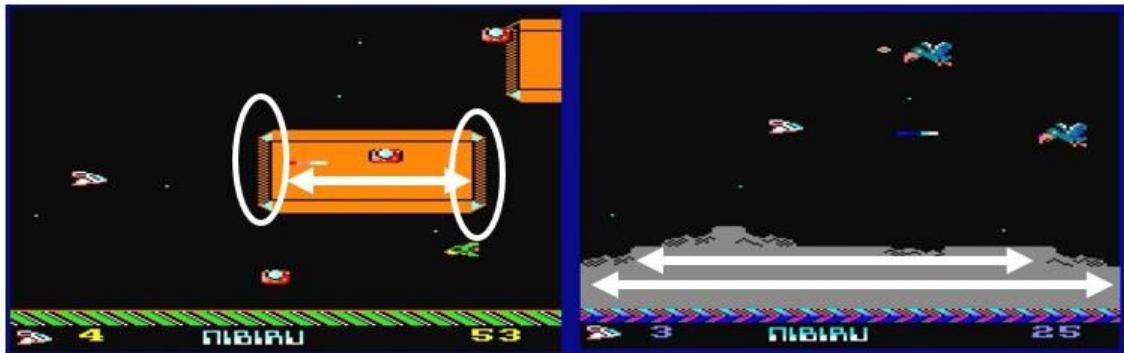


Εικ. 81 Βουνό που δημιουργήθηκε με 6 sprites και την τεχνική του "spotting".

Μετακινώντας όλα τα sprites προς τα αριστερά με αυτόματη ή σχετική κίνηση, τα sprites στα αριστερά θα αρχίσουν να "μοντζουρώνουν" το φόντο και επομένως

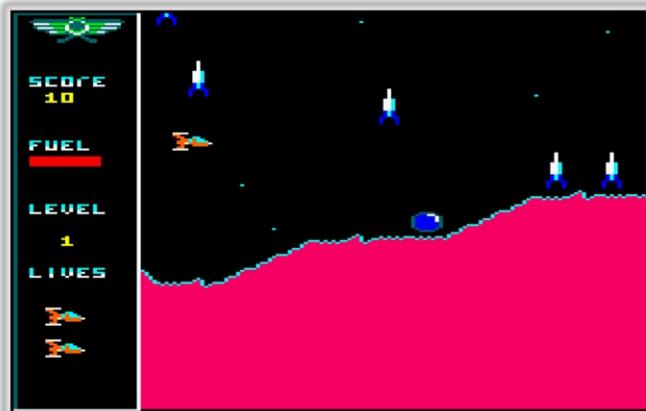
"γεμίζοντας" το βουνό, την ίδια στιγμή που τα sprites στα δεξιά θα αρχίσουν να το καθαρίζουν. Αν το βουνό εμφανίζεται αργά καθώς μπαίνει στην οθόνη, θα μοιάζει με ένα τεράστιο sprite βουνού, ενώ στην πραγματικότητα πρόκειται για 6 μικρά sprites.

Το βιντεοπαιχνίδι "Nibiru" χρησιμοποιεί την τεχνική "smearing" για να σχεδιάσει τα βουνά και άλλα μεγάλα στοιχεία, σε συνδυασμό με την εντολή MAP2SP που θα δούμε αργότερα.



Εικ. 82. Παραδείγματα της τεχνικής χρώσης

Χρησιμοποίησα επίσης την τεχνική της μουτζούρας στο βιντεοπαιχνίδι "Eridu", όπου τεράστια βουνά κινούνται ομαλά στην οθόνη.



Εικ. 83. Τεχνική χρώσης στο "Eridu".

Στην περίπτωση ενός παιχνιδιού κάθετης κύλισης, αν θέλουμε να ζωγραφίσουμε μια λίμνη σε ένα καφέ έδαφος, θα κάνουμε το ίδιο, κάποια sprites θα "λερώσουν" το έδαφος και άλλα πιο μακριά θα το "καθαρίσουν", κάνοντάς το να φαίνεται σαν μια τεράστια λίμνη, σε ένα κομμάτι.

Υπάρχει μόνο μια προφύλαξη που πρέπει να λάβετε, και αυτή είναι να μην πετάξουν τα πλοία πάνω από τη λίμνη, αλλιώς το "κόλπο" σας θα αποκαλυφθεί! Σε περίπτωση που θέλετε να είναι δυνατή η πτήση πάνω από τη λίμνη, τότε πρέπει να χρησιμοποιήσετε την αντικατάσταση sprite, όπως γίνεται στο στάδιο 2 του βιντεοπαιχνιδιού "Nibiru", όπου το σκάφος σας και τα εχθρικά σκάφη μπορούν να περάσουν πάνω από μεγάλα χρωματιστά ορθογώνια χωρίς να τα καταστρέψουν.

14.4 MAP2SP: Κύλιση με βάση έναν παγκόσμιο χάρτη

Όλες οι προηγούμενες τεχνικές είναι απολύτως έγκυρες για κύλιση, και μάλιστα συμβατές με αυτό που θα δούμε τώρα, που είναι η θεμελιώδης τεχνική που θα σας επιτρέψει να σχεδιάσετε έναν "παγκόσμιο χάρτη" και να κάνετε τον χαρακτήρα σας ή το σκάφος σας να κυλήσει μέσα σε αυτόν, με μία μόνο γραμμή κώδικα.

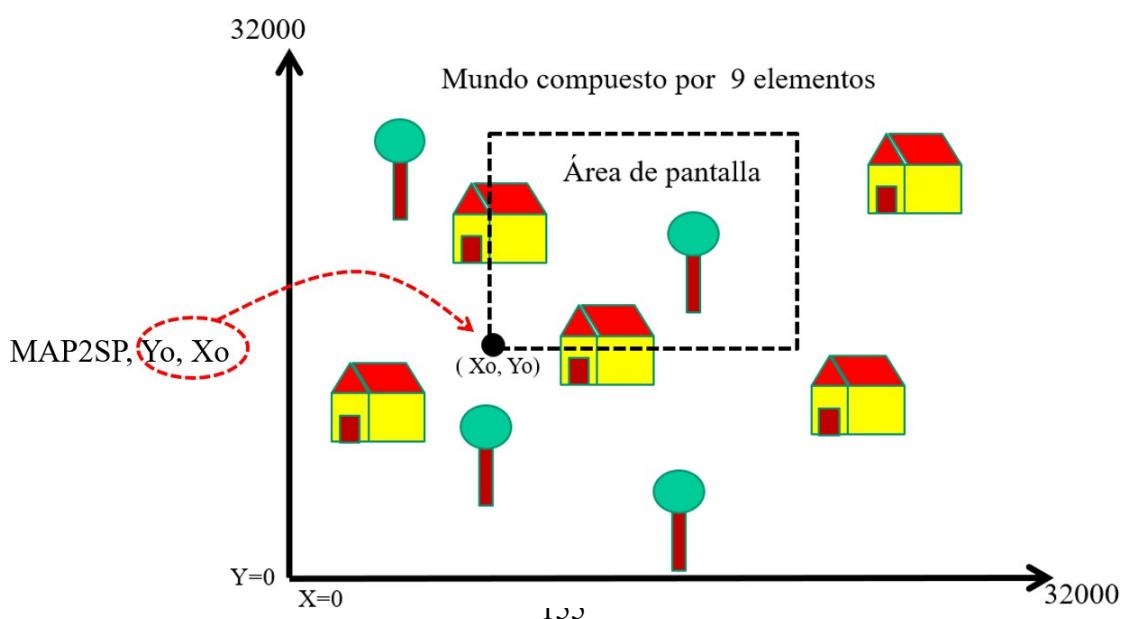
Για να χρησιμοποιήσετε την εντολή **|MAP2SP** είναι απαραίτητο να επιλέξετε την "**επιλογή συναρμολόγησης**" 2, η οποία μας δίνει 24,8 KB ελεύθερα για την καταχώριση BASIC.

Η ιδέα είναι απλή: Θα δημιουργήσουμε μια λίστα με στοιχεία που συνθέτουν το χάρτη του κόσμου (μέχρι 82 στοιχεία που θα ονομάσουμε "στοιχεία χάρτη"). Κάθε στοιχείο περιγράφεται από τις συντεταγμένες Y,X όπου βρίσκεται και τη διεύθυνση μνήμης όπου βρίσκεται η εικόνα του εν λόγω στοιχείου (ένα σπίτι, ένα δέντρο κ.λπ.). Η εικόνα που σχετίζεται με ένα στοιχείο χάρτη μπορεί να έχει οποιοδήποτε μέγεθος θέλετε. Οι συντεταγμένες κάθε στοιχείου θα είναι ένας θετικός ακέραιος αριθμός, από 0 έως 32000.

Μόλις δημιουργηθεί ο χάρτης, θα καλέσουμε τη συνάρτηση:

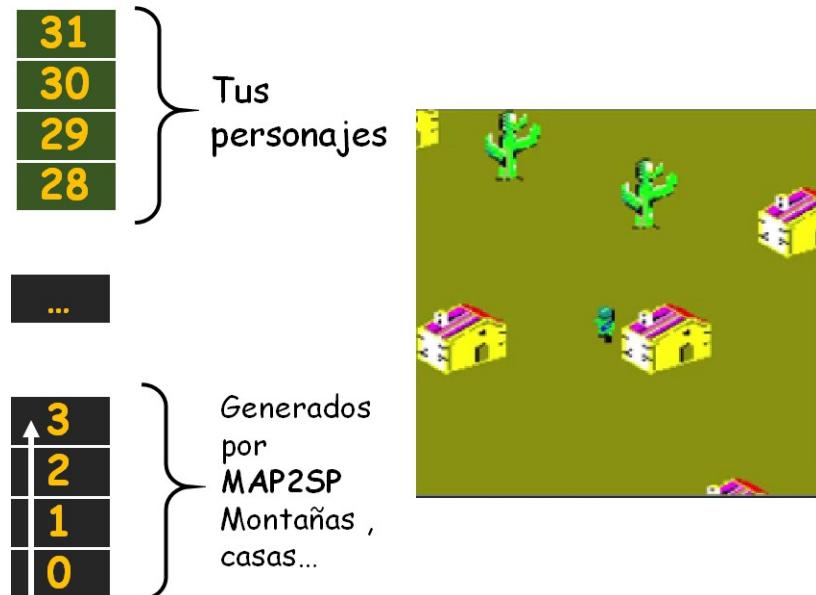
|MAP2SP, Yo, Xo

Αυτή η συνάρτηση αναλύει τη λίστα των στοιχείων στον κόσμο και καθορίζει ποια από αυτά εμφανίζονται αν ο κόσμος προβάλλεται τοποθετώντας την κάτω γωνία της οθόνης στις συντεταγμένες (I, Xo). Η συνάρτηση μετατρέπει τα "στοιχεία του χάρτη" σε sprites, καταλαμβάνοντας τις θέσεις στον πίνακα sprite από το μηδέν και μετά. Αυτό μπορεί να καταναλώσει πολλά ή λίγα sprites, ανάλογα με την πυκνότητα των στοιχείων χάρτη που έχετε. Σε μια επόμενη κλήση της ίδιας συνάρτησης, τα στοιχεία χάρτη που δεν υπάρχουν πλέον στη σκηνή δεν θα καταναλώνουν sprites στον πίνακα και άλλα στοιχεία χάρτη θα αναλάβουν τη θέση τους. Αυτό σημαίνει ότι **η συνάρτηση MAP2SP καταναλώνει έναν μεταβλητό και απροσδιόριστο αριθμό sprites, ανάλογα με τον αριθμό των αντικειμένων χάρτη που είναι ορατά στην οθόνη ανά πάσα στιγμή**. Στο παράδειγμα που ακολουθεί θα χρησιμοποιήσετε 3 sprites κατά την κλήση της λειτουργίας MAP2SP στις συντεταγμένες που αναφέρονται.



Σχ. 84 Χάρτης του κόσμου και MAP2SP

Αν χρησιμοποιήσετε αυτόν τον μηχανισμό, ο χαρακτήρας και οι εχθροί σας πρέπει να χρησιμοποιούν sprites από το 31 και κάτω, αποφεύγοντας έτσι πιθανές συγκρούσεις μεταξύ των sprites που χρησιμοποιούνται από τον μηχανισμό κύλισης και των χαρακτήρων σας. Αν κατά τύχη το MAP2SP συναντήσει περισσότερα από 32 αντικείμενα προς μετάφραση σε sprites, θα αγνοήσει αυτά που υπερβαίνουν το 32.



Σχήμα 85 sprites που καταναλώνονται από το MAP2SP

Πρέπει να καλείτε το MAP2SP σε κάθε κύκλο παιχνιδιού ή τουλάχιστον κάθε φορά που αλλάζετε τις συντεταγμένες του σημείου θέασης από το οποίο θέλετε να βλέπετε τον κόσμο. Το "συρόμενο" παράθυρο με το οποίο κυνηγάτε τις εικόνες που μετασχηματίζονται σε sprites μετράει πάντα αυτό που μετράει η οθόνη (80 bytes x 200 γραμμές) ανεξάρτητα από το πώς ορίζετε την εντολή SETLIMITS. Δηλαδή, **ακόμη και αν η εντολή SETLIMITS ορίσει μια πολύ μικρή περιοχή ζωγραφικής**, ό,τι έχει "κυνηγήσει" το παράθυρο των 80 bytes x 200 γραμμών θα μετατραπεί σε sprites.

Τα sprites που δημιουργούνται από το MAP2SP δημιουργούνται από προεπιλογή με κατάσταση 3, δηλαδή με ενεργή τη σημαία εκτύπωσης (το PRINTSPALL το εκτυπώνει) και με ενεργή τη σημαία σύγκρουσης (το COLSP θα συγκρουστεί με αυτό). Αν θέλετε τα sprites να δημιουργηθούν με άλλη κατάσταση, απλά καλείτε την εντολή MAP2SP μία φορά με μία μόνο παράμετρο που υποδεικνύει την κατάσταση με την οποία πρέπει να δημιουργηθούν τα sprites. Με μία μόνο κλήση αυτού του τύπου, η εντολή ρυθμίζεται για τις ακόλουθες κλήσεις με συντεταγμένες.

|MAP2SP, <status>, <status>, <status>, <status>, <status>, <status>, <status>, <status>.

Παράδειγμα:

|**Αυτή η ρύθμιση ρυθμίζει την εντολή MAP2SP να εκτυπώνεται αλλά να μην μπορεί να συγκρουστεί.**

Έχοντας αποσαφηνίσει την έννοια, ας δούμε αναλυτικά πώς καθορίζεται ο παγκόσμιος χάρτης και ένα παράδειγμα χρήσης της συνάρτησης MAP2SP.

14.4.1 Χάρτης του κόσμου (Πίνακας χαρτών)

Ο πίνακας στον οποίο θα καταχωρήσουμε όλα τα στοιχεία του χάρτη ονομάζεται MAP_TABLE και καθορίζεται σε ένα αρχείο .asm με όνομα **map_table_tujuego.asm**

Αυτός ο πίνακας περιέχει τα στοιχεία που καθορίζουν τις εικόνες του παγκόσμιου χάρτη για τα παιχνίδια κύλισης. Ο πίνακας συναρμολογείται στην ίδια διεύθυνση μνήμης με το LAYOUT, δηλαδή στη διεύθυνση 42040. Αυτό σημαίνει ότι η διάταξη και ο παγκόσμιος χάρτης δεν μπορούν να χρησιμοποιηθούν ταυτόχρονα, αλλά αυτό δεν αποτελεί πρόβλημα, αφού ένα παιχνίδι κύλισης δεν θα χρησιμοποιήσει τη διάταξη και αντίστροφα. Επίσης, ο περιορισμός είναι μόνο ότι δεν μπορούν να χρησιμοποιηθούν ταυτόχρονα, αλλά ένα παιχνίδι θα μπορούσε να έχει μια φάση όπου χρησιμοποιεί τη διάταξη και μια άλλη όπου κάνει κύλιση με βάση τον παγκόσμιο χάρτη.

Ο πίνακας εισόδου του παγκόσμιου χάρτη ξεκινά με 3 γενικές παραμέτρους (που καταλαμβάνουν συνολικά 5 bytes) και έναν κατάλογο "στοιχείων χάρτη", τα οποία περιγράφονται από 3 παραμέτρους το καθένα (x, y, κατεύθυνση εικόνας).

Η λίστα μπορεί να περιέχει έως και 82 στοιχεία, αλλά ο αριθμός των στοιχείων μπορεί να περιοριστεί από μία από τις καθολικές παραμέτρους. Η λίστα, το πολύ, καταλαμβάνει τα αρχικά 5 bytes + 82 στοιχεία x 6 bytes = 5+492=497 bytes. Αν βάζαμε ένα ακόμα στοιχείο, θα υπερβαίναμε τα 500Bytes που προορίζονται για το χάρτη (τα οποία είναι τα ίδια που προορίζονται για τη διάταξη).

Ο πίνακας ξεκινά με 3 παραμέτρους:

- Το μέγιστο ύψος οποιουδήποτε στοιχείου χάρτη
- Μέγιστο πλάτος οποιουδήποτε στοιχείου χάρτη (να εκφράζεται ως αρνητικός αριθμός)
- Αριθμός στοιχείων (μέγιστο 82)

Οι δύο πρώτες παράμετροι είναι σημαντικές για να ελέγξετε πότε ένα sprite μπορεί να εμφανίζεται εν μέρει στην οθόνη, καθώς η συνάρτηση MAP2SP δεν γνωρίζει ούτε βρίσκει το πλάτος και το ύψος κάθε εικόνας. Γνωρίζει μόνο πού βρίσκεται το στοιχείο του χάρτη και υποθέτοντας το μέγιστο πλάτος και ύψος, ελέγχει αν το στοιχείο αυτό μπορεί να εισέλθει στην οθόνη. Εάν ναι, δημιουργείται ένα sprite από το στοιχείο χάρτη. Εάν αυτές οι δύο παράμετροι είναι μηδενικές, η επάνω αριστερή γωνία του αντικειμένου χάρτη πρέπει να βρίσκεται εντός της οθόνης για να μετατραπεί το αντικείμενο σε sprite.

Κάθε στοιχείο είναι μια πλειάδα 3 παραμέτρων, της οποίας προηγείται το μνημονικό "DW":

DW Y, X, <εικόνα>

Ας δούμε ένα παράδειγμα του αρχείου map_table_tujuego.asm

ΠΙΝΑΚΑΣ ΧΑΡΤΗΣ

3 παράμετροι πριν από τον κατάλογο των "στοιχείων χάρτη".
dw 50; μέγιστο ύψος ενός sprite σε περίπτωση που περάσει από την κορυφή και πρέπει να ζωγραφίσετε ένα μέρος του.

dw -40; μέγιστο πλάτος ενός sprite σε περίπτωση αριστερόστροφου sprite (αρνητικός αριθμός)

db 82; αριθμός στοιχείων χάρτη.το πολύ να είναι 82

και από εδώ ξεκινούν τα στοιχεία dw

100,10,HOUSE; 1

dw 50,-10,CACTUS;2

dw 210,0,HOME;3

```
dw 200,20,CACTUS;4
dw 100,40,HOUSE;5
dw 160,60,HOUSE;6
dw 70,70,HOUSE;7
dw 175,40,CACTUS;8
dw 10,50,HOUSE;9
dw 250,50,HOUSE;10
dw 260,70,HOUSE;11
dw 260,70,HOUSE;11
dw 290,60,CACTUS;12
dw 180,90,HOUSE;13
dw 60,100,HOUSE;14
dw 60,100,HOUSE;14
```

...

Για να σχεδιάσετε τον κόσμο σας, σας συνιστώ να πάρετε ένα καρό τετράδιο και να σχεδιάσετε σε αυτό τα στοιχεία που θέλετε να έχει ο κόσμος σας. Κάθε τετράγωνο του σημειωματάριου μπορεί να αντιπροσωπεύει ένα σταθερό ποσό, όπως 8 pixel ή 25 pixel. Το θέμα είναι ότι θα πρέπει να αφιερώσετε χρόνο για να σχεδιάσετε τον κόσμο που θέλετε και τον τρόπο που θέλετε να τον διανύσετε. Για παράδειγμα, υπάρχουν παιχνίδια με πολλαπλές κατευθύνσεις τύπου "Gauntlet" και άλλα παιχνίδια κάθετης κύλισης όπως το Commando. Είναι δική σας επιλογή, αλλά σε κάθε περίπτωση κάντε το με χρόνο και υπομονή και το αποτέλεσμα θα αξίζει τον κόπο.

Κάθε φάση του παιχνιδιού σας μπορεί να είναι ένας χάρτης. Στο 8BP μπορείτε να αλλάξετε το χάρτη όποτε θέλετε χρησιμοποιώντας τις λειτουργίες POKE. Συνήθως χρησιμοποιώ 1KB από το χώρο μνήμης που χρησιμοποιεί το 8BP, για παράδειγμα από το 23000 στο 24000, για να αποθηκεύσω όλες τις φάσεις (χάρτες) του παιχνιδιού και κάθε φορά που μπαίνω σε μια φάση φορτώνω στη διεύθυνση 42040 τον αντίστοιχο χάρτη με PEEKing και POKEing. Δηλαδή, φτιάχνω το αρχείο του χάρτη μου στη διεύθυνση 23000 και καταλαμβάνει 1Kbyte, αφήνοντας 23 KB για το πρόγραμμα BASIC μου. Για να μην συνθλίβονται αυτές οι πληροφορίες του χάρτη από τη basic, θα πρέπει να κάνω ένα MEMORY 22999 στην αρχή του παιχνιδιού.

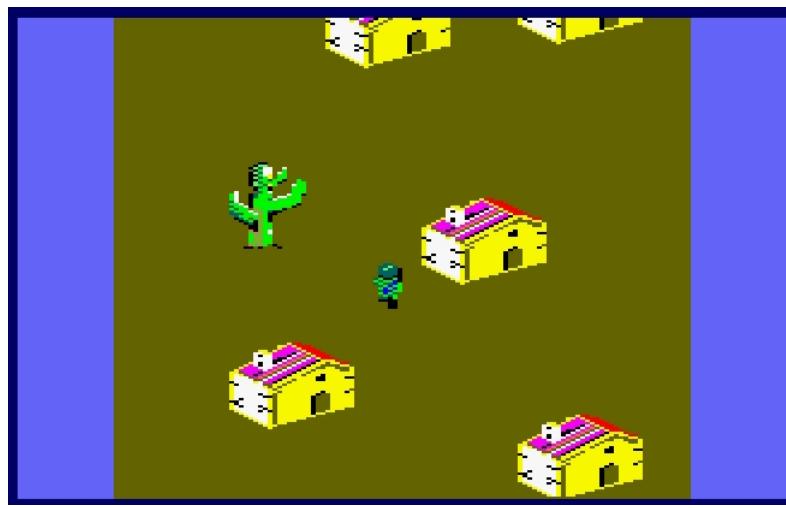
14.4.2 Χρήση της λειτουργίας MAP2SP

Ας δούμε τώρα ένα παράδειγμα χρήσης αυτής της συνάρτησης. Βασικά πρέπει να καλείται μία φορά σε κάθε κύκλο παιχνιδιού με τις νέες συντεταγμένες της αρχής από όπου παρατηρείται ο κόσμος.

Η συνάρτηση θα δημιουργήσει έναν μεταβλητό αριθμό sprites από το sprite 0 και μετά και θα τα δημιουργήσει με προσαρμοσμένες τις συντεταγμένες της οθόνης τους. Δηλαδή, ακόμη και αν ένα στοιχείο χάρτη έχει συντεταγμένες $x=100$, αν η αφετηρία κίνησης βρίσκεται στη θέση $x=90$, τότε το sprite αυτό θα δημιουργηθεί με συντεταγμένες οθόνης $x'=x-90=10$. Η συντεταγμένη του άξονα Y θα λάβει υπόψη ότι ο άξονας Y στο Amstrad αναπτύσσεται προς τα κάτω, ενώ ο παγκόσμιος χάρτης αναπτύσσεται προς τα πάνω. Έτσι, η συντεταγμένη Y προσαρμόζεται χρησιμοποιώντας την εξίσωση $Y'=200-(Y-Y_{orig})$. Άλλα μην ανησυχείτε, αυτή η προσαρμογή γίνεται ήδη από τη συνάρτηση MAP2SP. Απλά πρέπει να αλλάξετε την κινούμενη αφετηρία από όπου θα πρέπει να εμφανίζεται ο παγκόσμιος χάρτης.

Σε αυτό το μίνι-παιχνίδι, έχει δημιουργηθεί ένας κόσμος με σπίτια και κάκτους και ο χαρακτήρας μας περπατά ανάμεσα στα στοιχεία. Σε αυτό το παράδειγμα, σε περίπτωση

σύγκρουσης (που ανιχνεύεται με το **|COLSPALL**), ο χαρακτήρας δεν θα μπορέσει να συνεχίσει. Σε ένα παιχνίδι με αεροσκάφη όπου τα στοιχεία του χάρτη είναι "ιπτάμενα", θα μπορούσαμε να παραμετροποιήσουμε τη σύγκρουση ώστε να ανιχνεύει μόνο συγκρούσεις με εχθρούς και πυρά και όχι με στοιχεία φόντου, χρησιμοποιώντας **|COLSP, 32, <sprite_initial>, <sprite_final>**.



Εικ. 86 Μίνι παιχνίδι κύλισης εμπνευσμένο από το "commando".

Οπως μπορείτε να δείτε, είναι πολύ μικρό, αλλά τα έχει όλα: πολυκατευθυντική κύλιση, ανάγνωση πληκτρολογίου, αλλαγή ακολουθιών κινούμενων σχεδίων χαρακτήρων, ανίχνευση σύγκρουσης, μουσική...

ΣΗΜΑΝΤΙΚΟ: σημειώστε την εντολή MEMORY. Χρησιμοποιήσαμε την επιλογή συναρμολόγησης 2, ιδανική για παιχνίδια κύλισης, η οποία μας αφήνει σχεδόν 25 KB ελεύθερα.

```

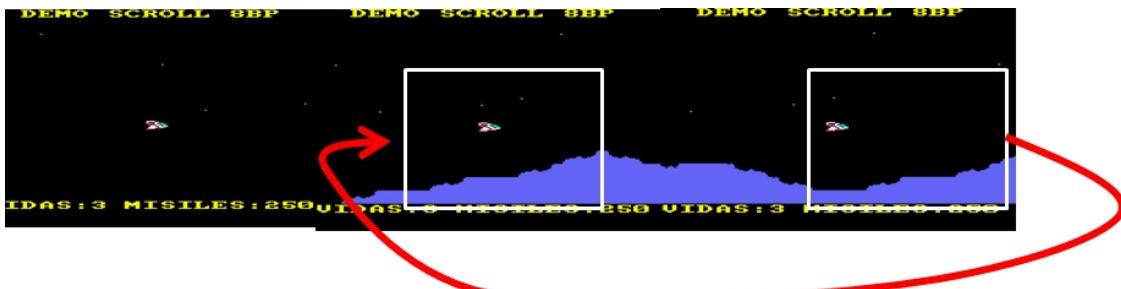
10 MNHMH 24799
20 MODE 0
30 ΣΤΟ ΔΙΑΛΕΙΜΜΑ GOSUB 280
40 CALL &6B78
50 DEFINT a-z
60 INK 0,12
70 FOR y=0 TO 400 BHMA 2
80 PLOT 0,y,10:DRAW 78,y
90 PLOT 640-80,y,10:DRAW 640,y
100 ΕΠΟΜΕΝΟ
110 x=0:y=0
120 |SETUPSP,31,0,&X100001
130 |SETUPSP,31,7,1:dir=1:' αρχική κατεύθυνση προς τα πάνω
140 |locatesp,31,100,36
150 |MUSIC,0,1,5
160 |SETLIMITS,10,70,0,199: |PRINTSPALL,0,1,0
170 col%=32:sp%=32:|COLSPALL,@sp%,@col%
180 |COLSP, 34, 0, 0, 0, 0: Σύγκρουση REM μόλις υπάρξει ελάχιστη επικάλυψη
190 'αρχίζει κύκλο παιχνιδιού
200 AN INKEY(27)=0 TOTE x=x+1:AN dir<>>3 TOTE
dir=3:|SETUPSP,31,7,3: GOTO 220
210 AN INKEY(34)=0 TOTE x=x-1:AN x<0 TOTE x=0:ELSE AN dir<>>4 TOTE
dir=4:|SETUPSP,31,7,4
220 AN INKEY(67)=0 TOTE y=y+2:AN x=xa KAI dir <> 1 TOTE
dir=1:|SETUPSP,31,7,1: GOTO 240
230 IF INKEY(69)=0 THEN y=y-2:IF y<0 THEN y=0:ELSE IF x=xa AND dir <>2
THEN dir=2:|SETUPSP,31,7,2:
240 EAN xa=x KAI ya=y TOTE dir=0 ELSE |ANIMA,31
250 |MAP2SP,y,x:|COLSPALL: IF col<32 THEN x=xa:y=ya:|MAP2SP,y,x ELSE
xa=x:ya=y
260 |PRINTSPALL

```

270 GOTO 200

280 |MUSIC:MODE 1: INK 0,0: PEN 1

Ας δούμε τώρα ένα άλλο παράδειγμα οριζόντιας κύλισης, όπου έχει επιτευχθεί ένα ενδιαφέρον εφέ ενός "άπειρου" παγκόσμιου χάρτη, κάνοντας το τέλος του χάρτη ίσο με την αρχή και προκαλώντας ένα απότομο άλμα όταν το Χο φτάσει μια συγκεκριμένη τιμή. Στην πραγματικότητα, αυτός ο παγκόσμιος χάρτης έχει μόνο 13 στοιχεία.



Eik. 87 Χάρτης των "άπειρου" κόσμου

Αυτός είναι ο χάρτης που χρησιμοποιήθηκε

_MAP_TABLE

3 παράμετροι πριν από τον κατάλογο των "στοιχείων χάρτη".
dw 50; μέγιστο ύψος ενός sprite σε περίπτωση που περάσει από
την κορυφή και ένα μέρος του πρέπει να ζωγραφιστεί.
dw -18- μέγιστο πλάτος οποιουδήποτε στοιχείου χάρτη. πρέπει να
εκφράζεται ως αρνητικός αριθμός.
db 13; αριθμός στοιχείων χάρτη. το πολύ να είναι 82

και από εδώ ξεκινούν τα στοιχεία dw

36,80,MONTUP; 1
dw 48,100,MONTUP;2
dw 60,120,MONTUP;3
dw 72,130,MONTUP;4
dw 72,140,MONTDW;5
dw 60,160,MONTH;6
dw 60,180,MONTDW;7
dw 48,190,MONTDW;8
dw 48,190,MONTDW;8

εδώ επαναλαμβάνω τα στοιχεία για να ταιριάζουν με τη θέση 100 dw 48,210,MONTUP;9

dw 60,230,MONTUP;10
dw 72,240,MONTUP;11
dw 72,250,MONTDW;12
dw 60,270,MONTH;13
dw 60,270,MONTH;13

;-----

Και αυτό είναι το πρόγραμμα BASIC, όπου έχω επισημάνει τη γραμμή όπου ο κόσμος πηγαίνει προς τα πίσω χωρίς ο παίκτης να καταλάβει τίποτα.

10 MNHMH 24799

11 FOR dir=42540 TO 42618 BHMA 2: POKE dir,20+RND*110:POKE dir+1,RND*80:NEXT

20 MODE 0

```

30 ΣΤΟ ΔΙΑΛΕΙΜΜΑ GOSUB 280
40 CALL &6B78
50 DEFINT a-z
51 INK 0,0
52 |MUSIC,0,0,0,0,5
110 xo=0:yo=0
111 x=36:y=100
120 |SETUPSP,31,0,&X100001
130 |SETUPSP,31,7,1:dir=1:' αρχική κατεύθυνση προς τα πάνω
140 |LOCATESP,31,y,x
160 |SETLIMITS,0,80,0,176: |PRINTSPALL,0,1,0
161 LOCATE 1,23 :PEN 1: PRINT "LIVES:3 MISSILES:250" PRINT "LIVES:3
MISSILES:250" PRINT "LIVES:3 MISSILES:250" PRINT "LIVES:3 MISSILES:250"
PRINT "LIVES:3 MISSILES:250" PRINT "LIVES:3 MISSILES:250"
162 LOCATE 1,1:PRINT " DEMO SCROLL 8BP"
170 col%=32:sp%=32:|COLSPALL,@sp%,@col%
180 |COLSP, 34, 0, 0, 0, 0: Σύγκρουση REM μόλις υπάρξει ελάχιστη επικάλυψη
190 'αρχίζει κύκλο παιχνιδιού
200 AN INKEY(27)=0 TOTE x=x+1: GOTO 220
210 AN INKEY(34)=0 TOTE x=x-1:AN x<0 TOTE x=0
220 AN INKEY(69)=0 TOTE y=y+2: GOTO 240
230 AN INKEY(67)=0 TOTE y=y-2:AN y<0 TOTE y=0
240 EAN xa=x KAI ya=y TOTE dir=0 ELSE |ANIMA,31
250 |MAP2SP,yo,xo:|COLSPALL:IF col<32 THEN END END
260 |PRINTSPALL
261 cycle=cycle +1: IF cycle=2 THEN |STARS,0,5,2,0,-1:ciclo=0
262 xo=xo+1:IF xo=210 THEN xo=100
263 |LOCATESP,31,y,x
270 GOTO 200
280 |MUSIC:MODE 1: INK 0,0:PEN 1

```

14.4.3 Παράδειγμα αρχείου φάσης

Αν θέλετε να έχετε πολλά στάδια σε ένα παιχνίδι κύλισης, όπως είπα προηγουμένως, μπορείτε να τα έχετε προεγκατεστημένα σε μια περιοχή μνήμης. Για παράδειγμα, μπορείτε να συγκεντρώσετε τα στάδια στη διεύθυνση 23000 και στη συνέχεια να έχετε 1000 bytes για να αποθηκεύσετε διάφορους παγκόσμιους χάρτες, αφού η 8BP ξεκινάει από τη διεύθυνση 24000. Σε αυτή την περίπτωση το παιχνίδι σας θα πρέπει να ξεκινήσει με την περιοχή MNHMHΣ 22999.

Το βιντεοπαιχνίδι "Nibiru" το κάνει αυτό, αν και δημιουργήθηκε όταν η 8BP ξεκίνησε στη διεύθυνση 26000 (δημιουργήθηκε με την 8BP v26), οπότε αποθηκεύει τον χάρτη από το 25000 και μετά. Για να φορτώσει μια φάση, απλά διαβάζει την περιοχή όπου έχει συναρμολογηθεί κάθε φάση και την γράφει πάνω από τη διεύθυνση όπου πρέπει να βρίσκεται ο πίνακας του κόσμου πριν αρχίσει να παίζει σε αυτή τη φάση. Αυτές οι τρεις γραμμές BASIC δείχνουν πώς να αντιγράψετε τη φάση 1 του παιχνιδιού (&61a8 = 25000)

```

310 ' βγάζει από τον παγκόσμιο χάρτη
320 dirmap!=42040:FOR i!=&61A8 TO &620D
330 dato=PEEK(i!):POKE dirmap!,dato:dirmap!=dirmap!+1
340 ΕΠΟΜΕΝΟ

```

Υπάρχει ένας ταχύτερος τρόπος να φορτώσετε τις φάσεις, χρησιμοποιώντας την

εντολή |UMAP, την οποία θα εξηγήσω αργότερα, αλλά αυτός ο βρόχος FOR με POKES είναι απολύτως έγκυρος.

Τέλος, στη συνέχεια, σας δείχνω το αρχείο stages του παιχνιδιού "Nibiru", το οποίο συναρμολογούμε στη διεύθυνση 25000 (θυμηθείτε ότι η έκδοση του 8bp με την οποία δημιουργήθηκε το "Nibiru" ξεκίνησε από τη διεύθυνση 26000, οπότε από τη διεύθυνση 25000 έως τη διεύθυνση 26000 υπήρχαν 1000 bytes ελεύθερα. Με την τρέχουσα έκδοση του 8BP θα συναρμολογούσαμε τις φάσεις χωρίς να φτάσουμε στη διεύθυνση 24800).

```
org 25000
ΦΑΣΗ1
;=====
_START_PHASE1
3 παράμετροι πριν από τον κατάλογο των "στοιχείων χάρτη".
dw 50; μέγιστο ύψος ενός sprite σε περίπτωση που περάσει από την
κορυφή και ένα μέρος του πρέπει να ζωγραφιστεί.
dw -18- μέγιστο πλάτος οποιουδήποτε στοιχείου χάρτη. πρέπει να εκφράζεται ως
αρνητικός αριθμός.
db 16; num items dw
36,82,MONTUP; 1 dw
48,104,MONTUP;2 dw
60,126,MONTUP;3 dw
72,138,MONTUP;4 dw
72,150,MONTDW;5 dw
60,172,MONTH;6 dw
60,194,MONTDW;7 dw
48,206,MONTDW;8 dw
60,172,MONTH;6 dw
60,194,MONTDW;7 dw
48,206,MONTDW;8
εδώ επαναλαμβάνω τα στοιχεία για να ταιριάζουν με τη θέση
100 dw 48,228,MONTUP;9
dw 60,250,MONTUP;10
dw 72,262,MONTUP;11
dw 72,274,MONTDW;12
dw 60,296,MONTH;13
dw 60,320,MONTDW;14
dw 48,350,MONTDW;15
dw 36,380,MONTDW;16
dw 36,380,MONTDW;16
_END_PHASE1
;=====
ΦΑΣΗ2
;=====
_START_PHASE2
dw 50; μέγιστο ύψος ενός sprite σε περίπτωση που περάσει από την
κορυφή και πρέπει να ζωγραφίσετε ένα μέρος του.
dw -6- μέγιστο πλάτος οποιουδήποτε στοιχείου χάρτη. πρέπει να εκφράζεται ως
αρνητικός αριθμός
db 15
dw 128,80,PLACA2_L_OV
dw 128,110,PLACA2_R_OV
dw 192,116,PLACA2_L_OV
dw 192,126,PLACA2_R_OV
dw 92,130,PLACA_L_OV dw
92,150,PLACA_R_OV dw
124,151,PLACA_L_OV dw
124,171,PLACA2_R_OV dw
128,200,PLACA2_L_OV dw
128,210,PLACA2_R_OV dw
92,220,PLACA2_L_OV dw
92,230,PLACA2_R_OV dw
164,240,PLACA2_L_OV dw
164,260,PLACA2_R_OV dw
156,254,PLACA2_R_OV dw
156,254,CUPULA2_OV
```

```

_END_PHASE2
=====
ΦΑΣΗ3
=====
_START_PHASE3
dw 50; μέγιστο ύψος ενός sprite σε περίπτωση που περάσει από την κορυφή
και πρέπει να ζωγραφίσετε ένα μέρος του.
dw -80- μέγιστο πλάτος οποιουδήποτε στοιχείου χάρτη. πρέπει να εκφράζεται ως
αρνητικός αριθμός.
db 4
dw 40,0,MAR; 1
dw 40,80,SEA; 2
dw 189,0,CLOUDS; 2
dw 189,80,CLOUDS; 2
_END_PHASE3

```

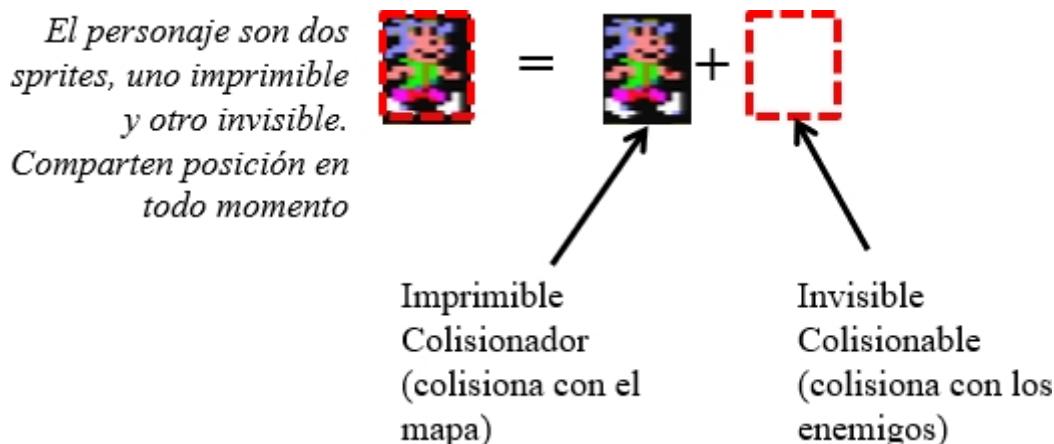
14.4.4 Σύγκρουση εχθρού με χάρτη

Ίσως θελήσετε να φτιάξετε ένα παιχνίδι όπου ο χαρακτήρας σας συγκρούεται με το χάρτη (χρησιμοποιώντας το COLSPALL) και επομένως είναι ένας συγκρουόμενος. Ας πούμε ότι έχετε αυτό:

- Ο χαρακτήρας σας: collider
- Στοιχεία χάρτη: συγκρουόμενα
- Η βολή σας: collider
- Εχθροί: συγκρουόμενοι

Τώρα, αν οι εχθροί σας είναι colliders, τότε δεν μπορούν να συγκρουστούν με το χάρτη, και μπορεί να θέλετε να φτιάξετε ένα παιχνίδι με γκάνγκστερ, για παράδειγμα. Η λύση είναι να κάνετε τους εχθρούς συγκρουόμενους. Άλλα φυσικά, σε αυτή την περίπτωση δεν μπορούν να σας σκοτώσουν πια. Και υπάρχει επίσης ένα πρόβλημα με το να είναι και ο πυροβολισμός collider.

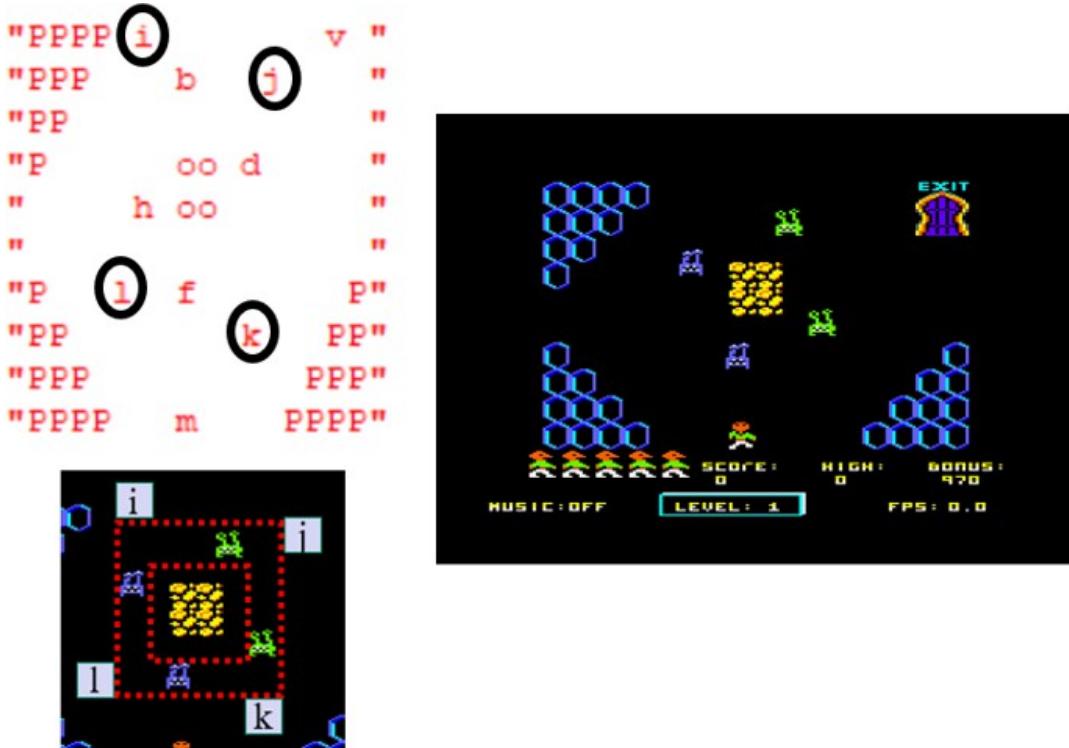
Η λύση σε αυτό το πρόβλημα βασίζεται στη χρήση ενός απλού τεχνάσματος που βασίζεται στα μη εκτυπώσιμα "άόρατα" sprites, τα οποία δεν ξοδεύουν CPU για να εκτυπωθούν αλλά είναι εκεί. Ο χαρακτήρας σας θα ξοδεύει δύο sprites: ένα εκτυπώσιμο sprite σύγκρουσης και ένα μη εκτυπώσιμο sprite σύγκρουσης, αλλά του ίδιου μεγέθους. Με τις βολές κάνουμε το ίδιο, ώστε να μπορούν να συγκρούονται με τους εχθρούς και το χάρτη.



Ένα παρόμοιο τέχνασμα χρησιμοποιείται στο παιχνίδι "Eternal Frogger".

Χρησιμοποιώντας αόρατα sprites, ο βάτραχος σκοτώνεται πέφτοντας στο ποτάμι (δείτε την ενότητα που εξηγεί την εντολή COLSPALL). Αυτό το απλό τέχνασμα μπορεί επίσης να εφαρμοστεί στις βολές του

χαρακτήρα. Τα "αόρατα" sprites είναι πολύ χρήσιμα. Στο παιχνίδι "Happy Monty" χρησιμοποιούνται για να κάνουν τους εχθρούς να αλλάζουν κατεύθυνση, έτσι ώστε όταν ένας εχθρός συγκρούεται με ένα αόρατο sprite, να αλλάζει κατεύθυνση. Αυτό σήμαινε ότι δεν ήταν απαραίτητο να οριστούν πολλές διαδρομές προσαρμοσμένες σε κάθε οθόνη, αλλά μόνο να τοποθετηθεί σε κάθε επίπεδο μια σειρά από αόρατα sprites που επέτρεπαν την αλλαγή των τροχιών των εχθρών (βλ. έγγραφο "making off" του Happy Monty).



14.4.5 Εικόνες φόντου στην κύλιση σας

Οι εικόνες φόντου προορίζονται για παιχνίδια κύλισης και είναι ένα χαρακτηριστικό που παρέχεται από την 8BP από την έκδοση V42 και μετά. Πρόκειται για έναν τύπο διαφανούς εκτύπωσης, κατά τον οποίο τα sprites που συνθέτουν το φόντο (ο παγκόσμιος χάρτης) μπορούν να εκτυπωθούν κάτω από τον χαρακτήρα και τους εχθρούς σας χωρίς να προκαλούν τρεμοπαίξιμο.

Οι εικόνες φόντου κάθε φορά που εκχωρούνται σε ένα Sprite εκτυπώνονται με αυτή την ειδική διαφάνεια και δεν έχει σημασία αν το Sprite έχει εκχωρήσει τη σημαία διαφάνειας στο byte κατάστασης ή όχι. Δείτε το κεφάλαιο 8 για να μάθετε πώς να τις χρησιμοποιείτε.

ΣΗΜΑΝΤΙΚΟ: Στον παγκόσμιο χάρτη μπορείτε να συνδυάσετε κανονικές εικόνες με "εικόνες φόντου" (ενότητα 8.5). Οι εικόνες φόντου έχουν πάντα διαφάνεια. Η σημαία διαφάνειας που χρησιμοποιείτε στο **|MAP2SP, <status>** θα ισχύει μόνο για κανονικές εικόνες.

ΣΗΜΑΝΤΙΚΟ: οι εικόνες φόντου είναι ακριβές στην εκτύπωση, και αν τις γυρίσετε,

είναι ακόμη πιο ακριβές. Αν το παιχνίδι σας έχει κύλιση, προσπαθήστε να τις χρησιμοποιήσετε για στοιχεία πάνω από τα οποία θα πετάξει ο χαρακτήρας ή οι εχθροί σας. Για παράδειγμα, για να επιταχύνετε την κύλιση χρησιμοποιήστε

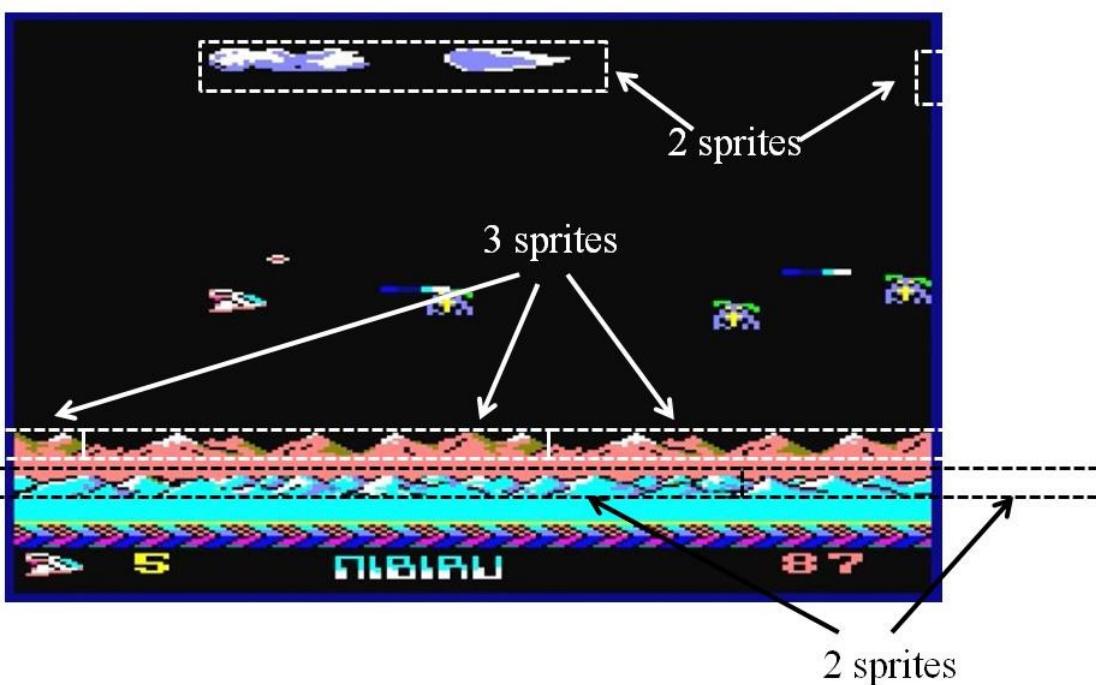
κανονικές εικόνες σε σπίτια ή βράχους που εμφανίζονται στις πλευρές της κύλισης, οι οποίες δεν θα επικαλύπτονται συχνά από τα sprites

14.5 Παράλλαξη κύλισης

Ας δούμε πώς μπορείτε να κάνετε μια κύλιση με παράλλαξη, δηλαδή με διάφορα "επίπεδα" σε διαφορετικές ταχύτητες. Ίσως μπορείτε να σκεφτείτε έναν άλλο τρόπο για να το κάνετε, αυτός είναι απλώς ένας πιθανός τρόπος, που χρησιμοποιείται στο παιχνίδι **"Nibiru"**.

Το πρώτο πράγμα που πρέπει να γνωρίζετε είναι ότι είναι πολύ πιο γρήγορο να εκτυπώσετε ένα πολύ μεγάλο οριζόντιο sprite από πολλά μικρά sprites. Αυτό οφείλεται στις λειτουργίες που πρέπει να εκτελεστούν μετά την ολοκλήρωση της ζωγραφικής κάθε γραμμής σάρωσης ενός sprite. Για τον ίδιο λόγο είναι πολύ πιο γρήγορο να εκτυπώσετε ένα πολύ μεγάλο οριζόντιο sprite από ένα κατακόρυφο sprite του ίδιου μεγέθους.

Θα τοποθετήσουμε δύο γιγαντιαία sprites στον παγκόσμιο χάρτη για να δημιουργήσουμε το νερό. Έφτιαξα ένα 160 x 8, οπότε έβαλα δύο από αυτά, ώστε όταν κάνει κύλιση, να αρχίσει να εμφανίζεται το επόμενο. Όταν η συνάρτηση MAP2SP κάνει όλο το γύρο της οθόνης, επιστρέφει στο x=0 και το όλο πράγμα επαναλαμβάνεται επ' άπειρον. Στον παγκόσμιο χάρτη έβαλα επίσης δύο sprites για τα σύννεφα.



Εικ. 88 κύλιση παράλλαξης

Για τα βουνά χρησιμοποίησα 3 κανονικά sprites, εκτός του παγκόσμιου χάρτη. Τους έδωσα αυτόματη κίνηση με βάση τη σημαία της κατάστασής τους και τα έκανα να κινούνται προς τα αριστερά. Όμως σε περιπτούς κύκλους απενεργοποιώ τόσο τη σημαία εκτύπωσης όσο και τη σημαία αυτόματης κίνησης, έτσι ώστε να κινούνται και να εκτυπώνονται μόνο κάθε δεύτερο κύκλο παιχνιδιού, επιτυγχάνοντας ταχύτητα μισή από αυτή του νερού. Τα sprites των βουνών είναι τα 16,17 και 18 και με αυτά τα σπρέι ενεργώ στο status byte τους.

**mc=cycle AND 1: IF mc THEN POKE 27256,0: POKE 27272,0: POKE 27288,0
ELSE POKE 27256,11: POKE 27272,11: POKE 27288,11**

Στο παράδειγμα "mc" είναι η μεταβλητή που καθορίζει αν ο κύκλος είναι μονός ή ζυγός.

14.6 Δυναμική ενημέρωση χάρτη: |UMAP

Ίσως στο παιχνίδι μας να χρειαζόμαστε έναν χάρτη με περισσότερα από 82 στοιχεία. Η απλά θέλουμε η εντολή |MAP2SP να εκτελείται γρηγορότερα χρησιμοποιώντας έναν μικρότερο χάρτη που ενημερώνουμε περιοδικά. Ή θέλουμε και τα δύο!

Για το σκοπό αυτό, από την έκδοση 32 της 8BP, υπάρχει η εντολή **UMAP** (συντομογραφία για "UPDATE MAP"). Αυτή η εντολή ενημερώνει τον χάρτη με πληροφορίες που βρίσκονται σε μια άλλη περιοχή μνήμης όπου έχουμε έναν μεγαλύτερο χάρτη. Η εντολή προκαλεί την πλήρη ανοικοδόμηση του χάρτη, περιλαμβάνοντας μόνο τα στοιχεία που ανταποκρίνονται σε συγκεκριμένες περιοχές συντεταγμένων X, Y (όλες οι παράμετροι είναι αριθμοί 16-bit).

Το UMAP δεν είναι μια απλή αντιγραφή στοιχείων. Είναι μια "επιλεκτική" αντιγραφή. Για παράδειγμα, αν έχουμε έναν χάρτη που βρίσκεται στη διεύθυνση 22500 που καταλαμβάνει 1500bytes και θέλουμε να ενημερώσουμε τον χάρτη με τις συντεταγμένες του χαρακτήρα μας, με αρκετό περιθώριο για να προχωρήσουμε στη συντεταγμένη Y μέχρι 100 γραμμές και στη συντεταγμένη x μέχρι 20 bytes προς όλες τις κατευθύνσεις:

| UMAP, 22500,23999, y-100, y+100, x-20, x+20

Αυτή η εντολή θα ελέγξει τις συντεταγμένες των στοιχείων που βρίσκονται στο χάρτη στη διεύθυνση 22500 και αν βρίσκονται εντός των περιθωρίων X, Y που έχουμε ορίσει, θα αντιγραφούν στην περιοχή μνήμης που χρησιμοποιεί η 8BP για την εντολή |MAP2SP, δηλαδή θα τα αντιγράψει από τη διεύθυνση 42040. Ωστόσο, θα αντιγράψει μόνο αυτά που πληρούν τη συνθήκη. Καθώς τα στοιχεία είναι λιγότερα, η εντολή |MAP2SP θα εκτελεστεί ταχύτερα, καθώς θα πρέπει να διαβάσει και να ελέγξει αν υπάρχουν λιγότερα στοιχεία στην οθόνη.

Mapa completo
Dirección 23000 (por ejemplo)



Mapa parcial ubicado
en la dirección 42040

UMAP, 23000,24500,200,500,0,80



MAP2SP,y,x

Sprites en
pantalla

MAP2SP se invoca en cada ciclo de juego. Cuando nos acerquemos a la frontera del mapa parcial invocationemos UMAP para que actualice el mapa parcial

Σχ. 89 UMAP

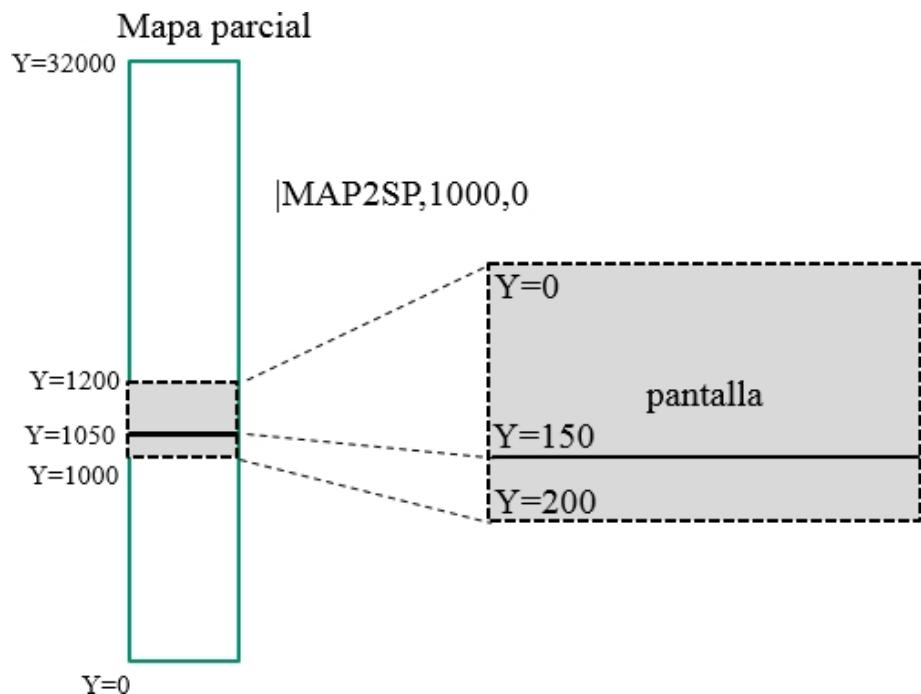
Σε τακτά χρονικά διαστήματα (όχι σε κάθε κύκλο παιχνιδιού) μπορούμε να ενημερώνουμε τον χάρτη με το UMAP και έτσι μπορούμε να δημιουργήσουμε πολύ μεγάλους κόσμους με πολλά στοιχεία και ταυτόχρονα θα έχουμε μεγαλύτερη ταχύτητα στο MAP2SP. Η εντολή UMAP είναι πολύ γρήγορη, αλλά η επίκλησή της σε κάθε κύκλο παιχνιδιού δεν έχει νόημα καθώς το MAP2SP μπορεί να δουλέψει με πολύ μεγαλύτερο χάρτη από αυτόν που χωράει στην οθόνη και μπορούμε να επικαλεστούμε την UMAP μόνο όταν χρειαζόμαστε περιοχές του αρχικού χάρτη που δεν υπάρχουν στον μερικό χάρτη. Το έχω χρησιμοποιήσει στο παιχνίδι αγώνων αυτοκινήτων "**3D Racing one**", καθώς ο χάρτης της πίστας ήταν πολύ μεγάλος (ξεπερνούσε τα 82 στοιχεία) και αυτό που έκανα ήταν να επικαλούμαι το |UMAP περιοδικά καθώς το αυτοκίνητο προχωράει.

Στη διεύθυνση πλήρους χάρτη (στο παράδειγμα 22500), θα έχουμε μόνο μια λίστα στοιχείων. **Δεν θα έχουμε τα 3 αρχικά δεδομένα που** περιέχονται στο χάρτη 42040 (εννοώ το μέγιστο ύψος κάθε στοιχείου του χάρτη, το μέγιστο πλάτος κάθε στοιχείου του χάρτη και τον αριθμό των στοιχείων). Ο αριθμός των στοιχείων ενημερώνεται από το |UMAP (ανάλογα με το πόσα στοιχεία πληρούν τα επιβαλλόμενα περιθώρια). Οι άλλες δύο παράμετροι καθορίζονται από εσάς στο αρχείο "map_table_your_game.asm".

Η εντολή |UMAP τοποθετεί τα στοιχεία στον μερικό χάρτη με μια σειρά που αποσκοπεί στην ταξινόμηση του μερικού χάρτη σύμφωνα με τη συντεταγμένη Y της οθόνης. Κανονικά επεξεργάζεστε τον συνολικό χάρτη σας σε ένα αρχείο που ονομάζεται "misupermapa.asm" ή κάτι τέτοιο. Και τον συναρμολογείτε στο 22500 (για παράδειγμα). Σε αυτό το αρχείο θα γράψετε ένα προς ένα τα στοιχεία του χάρτη σας, με αύξουσα σειρά της συντεταγμένης Y. Λοιπόν, για να πάρετε τα sprites ήδη ταξινομημένα με βάση τη συντεταγμένη Y (σε αύξουσα σειρά συντεταγμένων οθόνης), η εντολή |UMAP τα διαβάζει από το τέλος προς την αρχή. Με αυτόν τον τρόπο, τα sprites που θα δημιουργηθούν αργότερα με την |MAP2SP θα είναι ταξινομημένα με βάση τη συντεταγμένη Y της οθόνης. Θυμηθείτε ότι η οθόνη χρησιμοποιεί ένα αντίστροφο σύστημα συντεταγμένων σε σχέση με το χάρτη, δηλαδή η 150η συντεταγμένη της οθόνης είναι η 50η συντεταγμένη του χάρτη (στην περίπτωση της |MAP2SP,0,0). Αν δεν το καταλαβαίνετε αυτό πολύ καλά, μην ανησυχείτε. Είναι κάτι

από τις εσωτερικές λειτουργίες των **|UMAP** και **|MAP2SP** απλά για να είναι πιο αποδοτικές. Στο παρακάτω σχήμα έχω αναπαραστήσει το χάρτη και την οθόνη του CPC. Όπως μπορείτε να δείτε

ο χάρτης μπορεί να είναι πολύ μεγάλος, αλλά και οι συντεταγμένες του αυξάνονται προς τα πάνω, ενώ οι συντεταγμένες της οθόνης αυξάνονται προς τα κάτω.

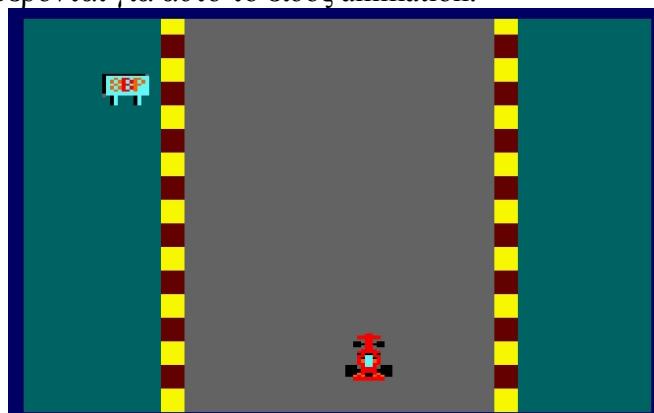


Eik. 90 MAP και οθόνη

14.7 Κινούμενα σχέδια και κύλιση μελανιού: Εντολή RINK

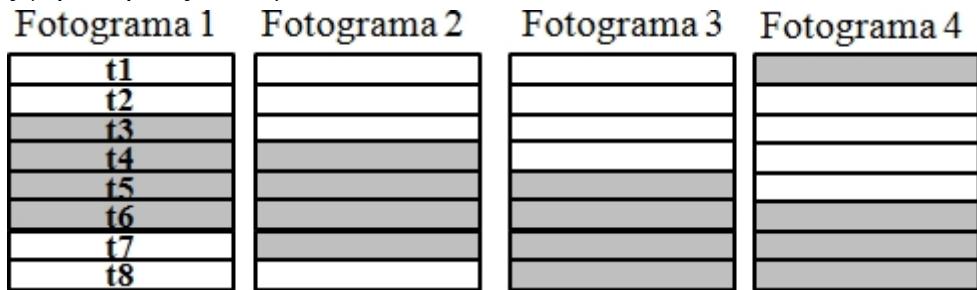
Υπάρχουν παιχνίδια που απαιτούν τη μετακίνηση μεγάλων μπλοκ μνήμης της οθόνης για να δώσουν την αίσθηση της κίνησης, όπως οι πλευρικές μπάρες σε παιχνίδια αγώνων ή μεγάλα μπλοκ από τούβλα ή χώμα. Η εντολή MAP2SP σας επιτρέπει να κάνετε όλα αυτά, αλλά η ταχύτητα δεν είναι αστραπιαία, επειδή ξοδεύει πολύ χρόνο της CPU για την κίνηση των sprites. Το Dye animation είναι το τέλειο συμπλήρωμα σε αυτές τις περιπτώσεις.

Σε υπολογιστές όπως ο AMSTRAD με την ισχυρή παλέτα των 16 ταυτόχρονων χρωμάτων, πολλά παιχνίδια χρησιμοποιούν την κίνηση με μελάνι. Ένα σαφές παράδειγμα είναι ορισμένα παιχνίδια αυτοκινήτων, των οποίων οι λωρίδες στην άκρη του δρόμου προσφέρονται για αυτό το είδος animation.



Εικ. 91 Λωρίδες με μελάνι

Η κινούμενη εικόνα με μελάνια συνίσταται στον ορισμό ενός συνόλου μελανιών πάνω στα οποία θα περιστρέφεται ένα σύνολο χρωμάτων. Ας δούμε ένα παράδειγμα με λευκές/γκρι λωρίδες και 8 μελάνια:



Eik. 92 Κίνηση μελανιού

Βασικά, για να δώσετε την αίσθηση της κίνησης, το πρώτο πράγμα που πρέπει να κάνετε είναι να αντιστοιχίσετε τα χρώματα στα μελάνια που πρόκειται να περιστραφούν. Στην προκειμένη περίπτωση τα χρώματα λευκό (26) και γκρι (26) και γκρι (26).

(13) αντιστοιχίζονται στα μελάνια t1..t8. Ας υποθέσουμε ότι το μελάνι t1 είναι 8, οπότε το μελάνι t8 θα είναι 15. Τα υπόλοιπα μελάνια (0 έως 7) θα χρησιμοποιηθούν για τα sprites. Σε κάθε χρονική στιγμή θα πρέπει να αναθέτουμε εκ νέου τις τιμές των 8 μελανιών για να δώσουμε την αίσθηση της περιστροφής. Αυτό κάνει η εντολή |RINK (συντομογραφία για "Rotate INK").

Το RINK σας επιτρέπει να ορίσετε ένα μοτίβο χρωμάτων που θα περιστρέφεται πάνω σε ένα σύνολο μελανιών. Ένα μελάνι δεν είναι χρώμα. Ένα μελάνι είναι ένα αναγνωριστικό στην περιοχή [0..15] που προσδιορίζει ένα χρώμα στην περιοχή [0..26]. Για να ορίσουμε το μοτίβο των χρωμάτων που θα περιστραφούν, θα χρησιμοποιήσουμε την εντολή RINK ως εξής:

RINK, <initial_ink>, <colour1>,<colour2>, <colour3>, ... ,<colourN>, <colourN>, ... ,<colourN>, <colourN>, <colourN>, <colourN>.

Αυτό υποδηλώνει ότι θα εναλλάσσουν N μελάνια ξεκινώντας από το αρχικό μελάνι χρησιμοποιώντας το υποδεικνύμενο χρωματικό μοτίβο. Σημειώστε ότι αν χρησιμοποιείτε πολλά μελάνια για να κάνετε ένα animation, θα σας μείνουν λιγότερα μελάνια για τα sprites σας.

Μόλις δημιουργηθεί το μοτίβο, μπορούμε να περιστρέψουμε τα μελάνια περισσότερο ή λιγότερο γρήγορα με

RINK, <βήμα>

Η τιμή βήματος είναι ο αριθμός των μετατοπίσεων που θα υποστούν τα μελάνια στο στένσιλ και, επομένως, μια υψηλότερη τιμή δίνει μεγαλύτερη ταχύτητα μετατόπισης.

Σύσταση: Λόγω της χρήσης των διακοπών RINK, η RINK μερικές φορές "τραυλίζει" όταν χρησιμοποιείται ταυτόχρονα με την εντολή |MUSIC στην ταχύτητα 6. Σε περίπτωση που θέλετε να χρησιμοποιήσετε και τις δύο ταυτόχρονα χωρίς παρεμβολές, χρησιμοποιήστε άλλη ταχύτητα για τη μουσική (μπορείτε να χρησιμοποιήσετε την ταχύτητα 5 ή 7, και οι δύο θα λειτουργήσουν άψογα).

14.7.1 2D αγωνιστικά αυτοκίνητα

Το παράδειγμα του παιχνιδιού αυτοκινήτων μπορείτε να το βρείτε στον ακόλουθο κατάλογο. Ο ήχος του κινητήρα έχει σκοπό να δώσει την αίσθηση της επιτάχυνσης. Για τις πινακίδες στα πλάγια έχει χρησιμοποιηθεί ένα sprite σχετικής κίνησης, το οποίο κινείται με την ίδια ταχύτητα με το βήμα των λωρίδων. Το μοτίβο μελανιού ορίζεται στη γραμμή 100.

| RINK,1,3,3,3,3,3,3,3,3,24,24,24,24,24: Rem κίτρινες και κόκκινες λωρίδες

```
1 MNMHM 24999
2 CALL &6B78
3 FOR i=0 TO 31:|SETUPSP,i,0,0,0,0:NEXT:|AUTOALL,0:|PRINTSPALL,0,0,0,0
4 |SETUPSP,31,9,16: |SETUPSP,31,0,1: vy=0
10 ΛΕΙΤΟΥΡΓΙΑ 0
20 DEFINT a-z
31 |LOCATESP,31,160,40: x=40
32 |SETUPSP,30,9,17: |SETUPSP,30,0,17:|LOCATESP,30,-20,10
40 CALL &BC02:'προεπιλεγμένη παλέτα
50 GOSUB 430
60 INK 0,13
70 INK 14,10
80 linestinta=3
90 rangotintas=8
91 ' Δημιουργία του σχεδίου μελάνης
100 |RINK,1,3,3,3,3,3,3,3,24,24,24,24,24,24,24,24: Rem κίτρινες και κόκκινες λωρίδες
101 |RINK,0
110 y=400
120 ' PAINT ROAD -----
121 tini=1
130 ΓΙΑ strips=1 ΕΩΣ 10
140 FOR t=tini TO rangotintas+tini-1
150 FOR j=1 TO linestinta
160 PLOT 0,y,14:DRAW 136,y
170 PLOT 140,y,t:DRAW 160,y
180 PLOT 480,y,t:DRAW 500,y
190 PLOT 504,y,14:DRAW 640,y
200 y=y-2
210 NEXT j
220 ΕΠΟΜΕΝΟ
240 ΕΠΟΜΕΝΕΣ λωρίδες
250 skipb=-16:xc=65: cosa=0
270 Κύκλος παιχνιδιού REM -----
293 AN saltob=-16 TOTE 296
294 EAN jumpb>0 TOTE thing=-jump ELSE thing=thing-1
295 IF thing<0 THEN |RINK,thing:vy=3*thing:posv=posv-3*thing:|OVERALL,-vy,0:IF
saltob <=0 THEN thing=2-saltob
296 |PRINTSPALL
351 cycle=cycle+1:IF posv>240 THEN posv=-30:|LOCATESP,30,posv,xc:IF xc=10
THEN xc=65 ELSE xc=10
361 E INKEY(27)=0 THEN E x<52 TOTE x=x+1:POKE 27499,x:GOTO 370
A A
N N
362 E INKEY(34)=0 THEN E x>21 TOTE x=x-1:POKE 27499,x
A A
N N
370 E INKEY(67)=0 THEN E jumpb<16 TOTE jumpb=jumpb+1:jump=jumpb/4
A A
N N
380 E INKEY(69)=0 THEN E jumpb>-16 TOTE jumpb=jumpb-1:jump=jumpb/4
A A
N N
390 SOUND 1 ,6000/(skip+17),1,15
GOTO 270
421 REM PALETA
```

430 INK 0 ,
440 INK 1 , 5
450 INK ,
460 INK ,
470 INK , 26
INK 5 , 0
490 INK ,
500 INK , 8
510 INK 8 , 10

```

520 INK  9 , 12
530 INK  10 ,
      INK
      ,
550 INK  , 0
560 INK  , 23
570 INK  , 0
580 INK  ,
590

```

ΕΠΙΣΤΡΟΦΗ

14.7.2 Brick Scroll

Σε αυτό το παράδειγμα θα συνδυάσουμε τη χρήση της κίνησης μελάνης με κύλιση με βάση το |MAP2SP. Χρησιμοποιώντας ink animation θα μετακινήσουμε ένα μοτίβο από τούβλα που έχουμε σχεδιάσει με ένα επαναλαμβανόμενο sprite, ενώ το κάστρο και το δέντρο θα μετακινηθούν από το |MAP2SP.

Η μετακίνηση των τούβλων θα ήταν μια τεράστια δουλειά αν γινόταν από την KME, οπότε αυτή η τεχνική επιτρέπει το "αδύνατο". Είναι μια πολύ ισχυρή τεχνική, αν τη χρησιμοποιήσετε με εφευρετικότητα.



Εικ. 93 Κύλιση με ταυτόχρονη χρήση της κίνησης μελάνης και του MAP2SP

Το τούβλο που χρησιμοποιείται είναι στην πραγματικότητα ένα sprite 8 χρωμάτων, με το σχέδιο που φαίνεται παρακάτω:

								0
8	9	10	11	12	13	14	15	

Και το μοτίβο μελανιού είναι 0,6,3,3,3,3,3,3,3,3,3,3,3,3,3,3. Για να το δημιουργήσετε, απλά εκτελέστε την εντολή:
|RINK,8,0,6,3,3,3,3,3,3

Ο χαρακτήρας (sprite 31) παραμένει στο κέντρο της οθόνης, έχοντας τη δυνατότητα να πηδά και να κινείται και προς τις δύο κατευθύνσεις. Για το συγχρονισμό της κίνησης των μελανιών και του MAP2SP, ορίζεται βήμα=2 για την εντολή |RINK, καθώς ένα byte είναι δύο pixel και το MAP2SP κινείται σε επίπεδο byte.

Είναι ενδιαφέρον να σημειωθεί πώς το πουλί (sprite 30) πρέπει επίσης να επηρεαστεί από την κίνηση, καθώς η ταχύτητά του πρέπει να προστεθεί ή να αφαιρεθεί από εκείνη του χαρακτήρα.

Όσον αφορά το χρώμα, δεδομένου ότι χρησιμοποιείται ένα μοτίβο 8 χρωμάτων και χρησιμοποιείται επίσης αντικατάσταση, μας μένουν 2 χρώματα για το φόντο (κάστρο, κλαδιά) και 3 χρώματα για τα sprites (χαρακτήρας και πουλί). Είναι εύκολο να τροποποιήσετε το πρόγραμμα ώστε να χρησιμοποιεί ένα μοτίβο περιστροφής 4 χρωμάτων και έτσι να έχετε 5 χρώματα για τα sprites, το οποίο είναι πιο λογικό.

Ο πλήρης κατάλογος παρατίθεται παρακάτω.

```

10 MNHMH 24999
11 ON BREAK GOSUB 5000
20 CALL &6B78
30 FOR i=0 TO 31:|SETUPSP,i,0,0,0:0:NEXT:|AUTOALL,1:|PRINTSPALL,0,1,0
40 |SETUPSP,31,7,1:|SETUPSP,31,0,65:|LOCATESP,31,130,36:"χαρακτήρας
50 |SETUPSP,30,7,7:|SETUPSP,30,0,157:|LOCATESP,30,50,80:
|SETUPSP,30,15,2: "πουλί
60 MODE 0:DEFINT a-z
80 CALL &BC02:'προεπιλεγμένη παλέτα
90 ΣΥΝΟΡΑ 10
100 ΜΕΛΑΝΙ 6,15:ΜΕΛΑΝΙ 7,15
110 ΜΕΛΑΝΙ 4,26:ΜΕΛΑΝΙ 5,26
120 ΜΕΛΑΝΙ 2,0:ΜΕΛΑΝΙ 3,0
130 INK 1,14
140 ' Δημιουργία του σχεδίου μελάνης
170 tini=8:|RINK,tini,0,6,3,3,3,3,3,3
200 |RINK,0
210 LOCATE 1,1:pen 4:PRINT "DEMO |RINK και |MAP2SP".
220 ' PAINT wall -----
230 |SETUPSP,29,9,23:'τούβλο
231 y=152
232 FOR row=1 TO 6
240 FOR brick=xini to 42 step 2:|PRINTSP,29,y,brick*2:next
241 xini=(xini-1) mod 2: y=y+8
242 επόμενος
390 dir=1:x=0:xp=80: cycle=40: stepy=2
400 |MUSIC,0,0,0,0,7
409 ' κύκλος παιχνιδιού -----
410 |AUTOALL:|PRINTSPALL
450 EAN INKEY(27)=0 THEN |RINK, stepy:x=x+1:|MAP2SP,0,x:|MOVER,30,0,-1:if jump=0 then IF dir=2 THEN dir=1:|SETUPSP,31,7,dir ELSE |ANIMA,31
460 AN INKEY(34)=0 TOTE |RINK,-stepy::x=x-1:|MAP2SP,0,x:av jump=0
τότε AN dir=1 TOTE dir=2:|SETUPSP,31,7,dir ELSE |ANIMA,31
471 EAN INKEY(67)+jump=0 THEN
jump=cycle:|SETUPSP,31,0,205:|SETUPSP,31,15,dir-1:|SETUPSP,31,7,31+dir
472 EAN cycle-jump=20 THEN jump=0:|SETUPSP,31,7,dir:|MOVER,31,5,0:
490 |PEEK,27483,@xp:IF xp<-20 THEN |LOCATESP,30,50,80:'το πουλί ξεκινά ξανά
501 κύκλος=κύκλος+1
502 AN xant=x TOTE |MAP2SP,0,32000
503 xant=x:' IF still THEN Δεν εκτυπώνω το κάστρο ώστε να μην
αναβοσβήνει
510 GOTO 410
5000 |MUSIC:CALL &BC02:pen 1:MODE 2:END

```

15 Παιχνίδια πλατφόρμας

Η δυσκολία του προγραμματισμού ενός παιχνιδιού πλατφόρμας έγκειται κυρίως στον σωστό χειρισμό της φυσικής των αλμάτων και των συγκρούσεων των πλατφορμών. Κάτι που μπορείτε να βρείτε στο βιντεοπαιχνίδι: "Φρέσκα φρούτα και λαχανικά" που διατίθεται στο 8BP.



Εικ. 94 Φρέσκα φρούτα και λαχανικά

Άλματα:

Βασικά για τη φυσική των αλμάτων αντί να χρησιμοποιήσω την εξίσωση του Νεύτωνα έχω ορίσει μια διαδρομή όπου το Vy ξεκινάει από το -5 και μειώνεται καρέ-καρέ. Όταν φτάσουμε στη θέση ζενίθ, η εικόνα αλλάζει έτσι ώστε να διαγράφεται στην κορυφή και η ταχύτητα Vy γίνεται θετική και αυξάνεται σταδιακά.

Στην ουσία, είναι σαν να εφαρμόζετε την εξίσωση του Νεύτωνα, αλλά χωρίς τους υπολογισμούς.

Έλεγχος εδάφους

Ενώ ο χαρακτήρας περπατάει, ελέγχω κάθε καρέ για να δω αν υπάρχει δάπεδο. Καθώς οι πλατφόρμες ανήκουν στον παγκόσμιο χάρτη, χρησιμοποιούν χαμηλά sprite identifiers (<10), οπότε αν με το COLSPALL ανιχνεύσω έναν αριθμό <10, υπάρχει έδαφος. Αν το αποτέλεσμα της ανιχνευσης είναι 32 (τα sprites πηγαίνουν από το 0 έως το 31), τότε δεν υπάρχει τίποτα και ο χαρακτήρας πρέπει να αρχίσει να πέφτει. Οι εχθροί δεν ανιχνεύουν τίποτα. Απλά περπατούν κατά μήκος προκαθορισμένων διαδρομών, όπου υποδεικνύεται πόσα βήματα πρέπει να κάνουν σε κάθε κατεύθυνση και μετά ξεκινούν πάλι από την αρχή. Από εκεί και πέρα, το sprite εκτελεί τη διαδρομή βήμα προς βήμα, καλώντας το |AUTOALL (το οποίο εσωτερικά ήδη καλεί το |ROUTEALL).

Συγκρούσεις πλατφορμών:

όταν ο χαρακτήρας βρίσκεται στο μονοπάτι πτώσης, τον μετακινώ (χωρίς εκτύπωση) προς τα κάτω κατά 5 μονάδες και ανιχνεύω τη σύγκρουση του sprite. Στη συνέχεια, τον μετακινώ (χωρίς εκτύπωση) 5 μονάδες προς τα πάνω. Εάν η σύγκρουση είναι 32, δεν υπάρχει σύγκρουση και συνεχίζω την πτώση του χαρακτήρα. Εάν η σύγκρουση είναι μικρότερη από 10, ο χαρακτήρας έχει συγκρουστεί με μια πλατφόρμα. Σε αυτή την περίπτωση μετακινώ τον χαρακτήρα έτσι ώστε να ταιριάζει απόλυτα με την αρχή της πλατφόρμας, οπότε: η θέση του κεφαλιού της κούκλας είναι "posy" η θέση των ποδιών είναι posy+26 επειδή ο χαρακτήρας μετράει 21 και προσθέτω 5 επειδή πέφτει (υπάρχει 5 της αυτοδιάλυσης), οπότε στην πραγματικότητα μετράει 26. Επειδή οι πλατφόρμες έχουν τοποθετηθεί σε πολλαπλάσια του 8, απλά κρατάω το υπόλοιπο της όλης διαίρεσης, το οποίο μπορώ να αποκτήσω με ένα AND 7 με λίγα λόγια, δύο πολύ καλά μελετημένες εντολές, αλλά μόνο δύο:

```
dy = (posy%+26) AND 7  
| MOVER,31,5-dy,0
```

Στη συνέχεια, αναθέτω την ακολουθία κινούμενου σχεδίου περπατήματος, η οποία δεν είναι η ακολουθία πτώσης και τα καρέ της είναι 21 καρέ ψηλά.

16 Ορδές εχθρών σε παιχνίδια κύλισης

Τα παιχνίδια κύλισης παίζονται συνήθως ως διαδοχικές ορδές εχθρών διαφορετικών τύπων και τροχιών, καθώς προχωράτε κάθετα ή οριζόντια.

Αν θέλατε ο χάρτης σας να έχει 10 ορδές σε διαφορετικές στιγμές της προόδου του χάρτη (ή του κύκλου του παιχνιδιού), θα μπορούσατε να χρησιμοποιήσετε 10 εντολές IF, αλλά θα ήταν πολύ αργή η εκτέλεση κάθε κύκλου (κάθε έλεγχος IF κοστίζει ένα χιλιοστό του δευτερολέπτου).

Η καλύτερη λύση είναι να διατηρήσετε δύο πίνακες, έναν για τη θέση της ορδής και έναν για τον τύπο της ορδής.

Ευρετήριο (Αύξων αριθμός παραγγελίας)	Nexthorda (Κύκλος στον οποίο πρέπει να εμφανίζεται)	Tipohorda (Τύπος εχθρών ορδής)
0	100	1 - αεροσκάφος
1	200	2 - πύραυλοι
		4 - ATIA
	320	3 - δράκοι

10 dim tipohorda(10)

20 typeforda(0)=1: typeforda(1)=2: typeforda(2)=4:

t y p e f o r d a (3)=3:

30 dim nexthorda(0)

40 nexthorda(0)=100:nexthorda(1)=200: nexthorda(2)=250 ...

50 index=0

Κύκλος παιχνιδιού 100 rem

110 κύκλος=κύκλος +1

120 AN κύκλος = nexthorda(index) TOTE GOSUB 500

...

**500 δημιουργία ορδή
ρουτίνας rem**

**510 on tipohorda(index) Πάμε
600,700,800**

**520 rem ρουτίνη δημιουργίας ορδή τύπ 4. Al final θα το πηγα
α γία ος κάνουμε ίνετε στο**

**600 rem ρουτίνη δημιουργίας ορδή τύπ 1. Al final θα το πηγα
1000 α γία ος κάνουμε ίνετε στο
Δείκτης=δείκτης+1**

**1010 RETURN ρουτίνη δημιουργίας ορδή τύπ 2. Al final θα το πηγα
α γία ος κάνουμε ίνετε στο**

**800 rem ρουτίνη δημιουργίας ορδή τύπ 3. Al final θα το πηγα
α γία ος κάνουμε ίνετε στο**

...

Αντί να χρησιμοποιείτε τον κύκλο στον οποίο θα πρέπει να εμφανίζεται, μπορείτε επίσης να κάνετε τη σύγκριση με τη θέση του χάρτη στην οποία βρίσκεστε, αυτό θα εξαρτηθεί από το πώς θέλετε να το κάνετε, αλλά με αυτή την ιδέα των δύο πινάκων έχετε ήδη τη γενική προσέγγιση για να οργανώσετε τις ορδές των εχθρών σας.

17 Επαναπροσδιορίσιμοι μίνι-χαρακτήρες: PRINTAT

Το σετ χαρακτήρων της Amstrad είναι ωραίο και καλά κατασκευασμένο. Ωστόσο, στη λειτουργία 0 έχουμε μόνο 20 χαρακτήρες πλάτους ανά γραμμή και εμφανίζονται πολύ "φαρδιοί", οπότε μερικές φορές δεν είναι κατάλληλοι για την εμφάνιση ορισμένων κειμένων ή σημείων σε ένα παιχνίδι. Επίσης, η εντολή PRINT είναι πολύ αργή, οπότε δεν συνιστάται να ενημερώνετε συχνά τους δείκτες, καθώς το παιχνίδι "σταματάει" ενώ εκτυπώνει, είναι μόνο μερικά χλιοστά του δευτερόλεπτου, αλλά είναι αισθητό.

Για το λόγο αυτό, από την έκδοση v31 της 8BP έχει προστεθεί η εντολή PRINTAT, η οποία μπορεί να εκτυπώσει μια σειρά χαρακτήρων χρησιμοποιώντας ένα νέο, μικρότερο σύνολο χαρακτήρων (τους ονομάζω "μίνι-χαρακτήρες"). Αυτή η νέα εντολή σας επιτρέπει να χρησιμοποιήσετε το μηχανισμό διαφάνειας των sprites, ώστε να μπορείτε να εκτυπώνετε χαρακτήρες με σεβασμό στο φόντο. Λειτουργεί ως εξής:

|PRINTAT, <σημαία διαφάνειας>, γ, x, @string

Παράδειγμα:

cad\$= "Hello".

|PRINTAT, 0, 100, 10, @cad\$



Εικ. 95 PRINTAT

Η εντολή |PRINTAT εκτυπώνει συμβολοσειρές χαρακτήρων και όχι αριθμητικές μεταβλητές, οπότε αν θέλετε να εκτυπώσετε έναν αριθμό (για παράδειγμα, τους πόντους στον πίνακα αποτελεσμάτων στο βιντεοπαιχνίδι σας) πρέπει να το κάνετε:

```
points=points+1  
cad$= str$(points)  
|PRINTAT,0,100,10, @cad$
```

Η εντολή |PRINTAT δεν επηρεάζεται από τα όρια αποκοπής που έχουν οριστεί με την εντολή

|SETLIMITS. Αυτό είναι το πιο λογικό, δεδομένου ότι συνήθως θα χρησιμοποιήσετε το PRINTAT για να εκτυπώσετε αποτελέσματα στους δείκτες σας, οι οποίοι θα βρίσκονται εκτός της περιοχής που ορίζεται από το |SETLIMITS.

Σε αντίθεση με την εντολή PRINT της BASIC, η εντολή |PRINTAT είναι αρκετά γρήγορη και μπορεί να χρησιμοποιηθεί για να ενημερώνετε συχνά τους δείκτες του παιχνιδιού σας.

Το PRINTAT χρησιμοποιεί ένα επαναπροσδιορισμένο αλφάριθμο, το οποίο μπορεί να περιέχει μια μειωμένη ή διαφορετική έκδοση των "επίσημων" χαρακτήρων της Amstrad. Από προεπιλογή, το 8BP παρέχει ένα μικρό αλφάριθμο που αποτελείται από αριθμούς, κεφαλαία γράμματα, λευκό διάστημα και ορισμένα σύμβολα. Δεν θα μπορείτε να χρησιμοποιήσετε έναν χαρακτήρα που δεν περιλαμβάνεται σε αυτό το σύνολο, όπως τα πεζά γράμματα. Οι χαρακτήρες αυτού του αλφαριθμού έχουν όλοι το

ίδιο μέγεθος: 4 pixel πλάτος x 5 pixel ύψος, δηλαδή 2 bytes x 5 γραμμές.

Αυτή η συμβολοσειρά περιέχει όλους τους χαρακτήρες που μπορείτε να χρησιμοποιήσετε με το σειριακό αλφάριθμο 8BP. Σημειώστε ότι δεν υπάρχουν πεζά γράμματα και πολλά σύμβολα λείπουν, αν και μπορείτε να δημιουργήσετε το δικό σας αλφάριθμο που τα περιέχει. Το τελευταίο σύμβολο είναι το ":".

"0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ!: ,."



Σχ. 96 προεπιλεγμένο σύνολο χαρακτήρων που είναι διαθέσιμο στο 8BP για χρήση με το PRINTAT

ΣΗΜΑΝΤΙΚΟ: Εάν προσπαθήσετε να εκτυπώσετε με την PRINTAT έναν χαρακτήρα που δεν υπάρχει στο δημιουργημένο αλφάριθμο, θα εκτυπωθεί ο τελευταίος χαρακτήρας στη λίστα χαρακτήρων, ο οποίος στο προεπιλεγμένο αλφάριθμο είναι η τελεία "".

Δημιούργησα τους χαρακτήρες χρησιμοποιώντας τα μελάνια 2 και 4, έτσι ώστε να μπορεί να χρησιμοποιηθεί η υπερεκτύπωση, καθώς τα χρώματα φόντου είναι 0 και 1 και με τη χρήση της υπερεκτύπωσης το μελάνι 2 πρέπει να είναι ίσο με το 3 και το 4 πρέπει να είναι ίσο με το 5 (δείτε το κεφάλαιο όπου εξηγώ την υπερεκτύπωση). Για να χρησιμοποιήσετε την υπερεκτύπωση, απλά θέστε τη σημαία διαφάνειας στην εντολή PRINTAT σε "1".

17.1 Δημιουργήστε το δικό σας αλφάριθμο μίνι χαρακτήρων

Οι χαρακτήρες που θα χρησιμοποιηθούν με την εντολή PRINTAT ορίζονται σε ένα αρχείο στον κατάλογο ASM και εισάγονται από το αρχείο "images.asm".

Ας δούμε ένα κομμάτι του αρχείου "images.asm" Θα βρούμε αυτές τις τρεις γραμμές:

αν δεν πρόκειται να χρησιμοποιήσετε την εντολή **PRINTAT**, αλλά μόνο τους χαρακτήρες Amstrad

Στη συνέχεια μπορείτε να σχολιάσετε τις ακόλουθες 3 γραμμές

APXIZEI ΤΟ ΑΛΦΑΒΗΤΟ
διαβάστε "alphabet_default.asm"
_END_ALPHABET

Το αλφάριθμο αποτελείται από μερικά δεδομένα και τις εικόνες κάθε χαρακτήρα. Όλα αυτά συγκεντρώνονται μέσα στην περιοχή μνήμης εικόνων 8BP. Το προεπιλεγμένο αλφάριθμο είναι λίγο πάνω από 400 bytes.

Το αρχείο alphabet_default.asm περιέχει το αλφάριθμο που περιλαμβάνει η 8BP ως πρότυπο. Μπορείτε να δημιουργήσετε το δικό σας αρχείο αλφαριθμού. Αυτό το αρχείο περιέχει 3 μεταβλητές που υποδεικνύουν το μέγεθος των χαρακτήρων, τους οποίους μπορείτε να σχεδιάσετε σε όποιο μέγεθος θέλετε. Από προεπιλογή έχω δημιουργήσει ένα αλφάριθμο πλάτους 2 bytes και ύψους 5 γραμμών, αλλά μπορείτε να αποφασίσετε να χρησιμοποιήσετε άλλο μέγεθος, ακόμα και να δημιουργήσετε γιγαντιαίους χαρακτήρες. Αν αλλάξετε το πλάτος ή το ύψος, πρέπει να αλλάξετε ανάλογα και τη

μεταβλητή ALPHA_SIZE στο αρχείο alphabet.asm.

**ALPHA_width db 2; πλάτος αλφαβήτου.όλα τα γράμματα έχουν το
ίδιο μέτρο ALPHA_height db 5; ύψος αλφαβήτου.όλα τα
γράμματα είναι το ίδιο ALPHA_span db 2*5**

Στη συνέχεια, βρίσκουμε τη συμβολοσειρά που υποδεικνύει τους έγκυρους χαρακτήρες που μπορούν να χρησιμοποιηθούν με την εντολή PRINTAT. Αυτοί οι χαρακτήρες είναι έγκυροι επειδή έχουν ένα σχετικό σχέδιο. Μετά από αυτή τη συμβολοσειρά, τα σχέδια όλων αυτών των χαρακτήρων βρίσκονται στη συμβολοσειρά κειμένου ALPHA_LIST

ALPHA_LIST

"0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ:!.;" ; χαρακτήρες που

δημιουργήθηκαν

db 0 ;το μηδενικό byte υποδεικνύει το τέλος της συμβολοσειράς χαρακτήρων της λίστας Alpha_list

Οι εικόνες κάθε χαρακτήρα στη συμβολοσειρά ALPHA_LIST παρουσιάζονται παρακάτω. Τις δημιουργησα με το εργαλείο SPEDIT. Ο πρώτος από αυτούς πρέπει να είναι ο πρώτος χαρακτήρας της συμβολοσειράς ALPHA_LIST, δηλαδή "0". Τα bytes που αντιστοιχούν σε αυτόν τον χαρακτήρα εμφανίζονται εδώ:

<pre>; χαρακτήρας 0 db 12 , 8 db 8 , 8 db 8 , 8 db 32 , 32 db 48 , 32</pre>	
---	--

Στη συνέχεια θα βρούμε ένα προσ ένα τα υπόλοιπα γράμματα, αριθμούς και σύμβολα που έχουν οριστεί. Με λίγη υπομονή μπορείτε να δημιουργήσετε το δικό σας σύνολο μίνι χαρακτήρων, οι οποίοι θα δώσουν περισσότερη προσωπικότητα στο βιντεοπαιχνίδι σας. Χρειάζεται να δημιουργήσετε μόνο αυτούς που πρόκειται να χρησιμοποιήσετε, και όσο λιγότεροι είναι, τόσο λιγότερη μνήμη θα χρησιμοποιήσετε στην περιοχή της εικόνας.

Να θυμάστε ότι αν προσπαθήσετε να εκτυπώσετε έναν χαρακτήρα που δεν έχετε δημιουργήσει, θα εκτυπωθεί ο τελευταίος από τους χαρακτήρες που ορίζονται στη λίστα ALPHA_LIST.

17.2 Προεπιλεγμένο αλφάβητο για τη λειτουργία 1

Το προεπιλεγμένο αλφάβητο της 8BP δημιουργείται σε MODE 0 και αν προσπαθήσετε να το χρησιμοποιήσετε σε MODE 1 δεν θα λειτουργήσει σωστά, επειδή είναι σχέδια που έχουν δημιουργηθεί σε mode 0. Από προεπιλογή η 8BP έρχεται επίσης με ένα αλφάβητο που λειτουργεί σε MODE 1. Απλά πρέπει να αλλάξετε μια γραμμή του images.asm. Αυτό το αλφάβητο είναι εμπνευσμένο από μια γραμματοσειρά που ονομάζεται "5th agent".

<pre>_APXIZEI_TO_ΑΛΦΑΒΗΤΟ διαβάστε "alphabet_default_mode1.asm" -END_ALPHABET</pre>

0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ:!.;

18 Ψευδό 3D

Στη δεκαετία του 1980 τα παιχνίδια αυτοκινήτων τύπου "Pole position" ήταν δημοφιλή. Έδιναν μια τρισδιάστατη αίσθηση, αλλά δεν έκαναν τους τρισδιάστατους υπολογισμούς, μόνο για το σχέδιο του δρόμου, και μερικές φορές ούτε καν αυτό, καθώς ήταν προσεγγίσεις που έδιναν μια καλή αίσθηση της ταχύτητας.

Στα μηχανήματα arcade, χρησιμοποιήθηκαν ειδικά τσιπ για την εκτέλεση του "sprite scaling", που έκανε τα sprites να αυξάνονται σε μέγεθος ομαλά, και τον υπολογισμό του δρόμου μέσω ενός ειδικού τσιπ (όπως το "sega Road chip"), το οποίο ήταν αποκλειστικά αφιερωμένο στη ζωγραφική του δρόμου με τις λωρίδες του. Στη συνέχεια, ο οδικός χάρτης προστέθηκε στον χάρτη sprite και συντέθηκε η τελική εικόνα. Αυτά τα τσιπ χρησιμοποιήθηκαν σε μηχανήματα arcade όπως το "Pole Position" και το "Space Harrier".



Εικ. 97 Pole position και Out Run (μηχανήματα arcade)

Το κοινό χαρακτηριστικό τόσο των τεχνικών λογισμικού (που χρησιμοποιούνται σε 8bit υπολογιστές) όσο και των τεχνικών υλικού (που χρησιμοποιούνται σε arcade μηχανές) είναι ότι οι καμπύλες είναι μια ψευδαίσθηση: ο δρόμος παραμορφώνεται ενώ τα βουνά στον ορίζοντα μετακινούνται για να δώσουν την αίσθηση μιας στροφής, αλλά δεν υπάρχει καμία στροφή. Τα αποτελέσματα ήταν συχνά πολύ πειστικά, αλλά οι καμπύλες ήταν στην πραγματικότητα μια ψευδαίσθηση.

Οι τεχνικές που χρησιμοποιούνταν στους υπολογιστές 8-bit ήταν πολύ διαφορετικές. Τα πάντα ήταν αποδεκτά, αρκεί ο παίκτης να πιστεύει ότι βρίσκεται σε πίστα αγώνων. Σε πολλές περιπτώσεις, χρησιμοποιήθηκε η περιστροφή του μελανιού για να δοθεί η αίσθηση της ταχύτητας. Πολλά παιχνίδια υπέφεραν από χαμηλό ρυθμό καρέ, κάτω από 5 fps. Μεταξύ των καλύτερων παιχνιδιών για το Amstrad ήταν το "3D grand Prix" (με χρήση λογισμικού κλιμάκωσης sprite σε συνδυασμό με animation μελάνης), το "Buggy boy" που βασίστηκε σε μια πολύ προηγμένη τεχνική προγραμματισμού του raster effect και κάποια άλλα όπως το "Crazy cars" ή το "Chase HQ".

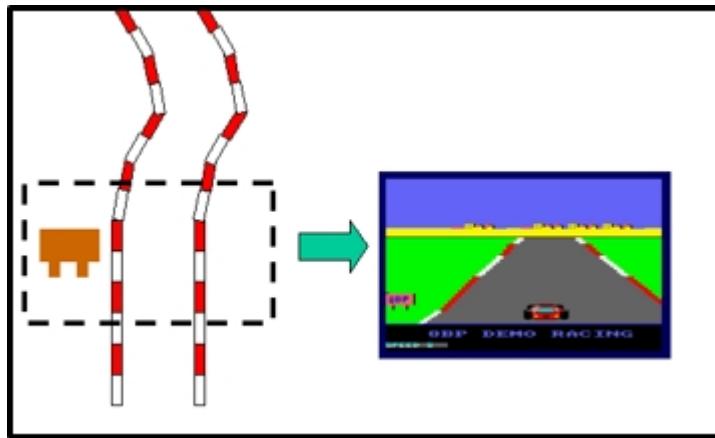
Από την έκδοση V32 του 8BP έχετε στη διάθεσή σας τη δυνατότητα ψευδο-3D, που χρησιμοποιείται στο παιχνίδι επίδειξης "3D Racing one". Είναι πολύ εύκολο στη χρήση και με αυτό μπορείτε να φτιάξετε τα δικά σας παιχνίδια αγώνων δρόμου, τανκς, πλοίων κ.λπ.

Για να χρησιμοποιήσετε το Pseudo-3D στην 8BP πρέπει να χρησιμοποιήσετε την "**επιλογή συναρμολόγησης**" 3, η οποία αφήνει 24 KB ελεύθερα για το πρόγραμμα BASIC.

Οι δυνατότητες που παρέχονται από το 8BP για τη διάθεση ψευδο-3D είναι οι εξής:

- 1) **Τρισδιάστατη προβολή:** πρόκειται για την προβολή ενός δισδιάστατου χάρτη του κόσμου σε τρισδιάστατη μορφή, έτσι ώστε, ακόμη και αν τον διασχίσουμε σε δισδιάστατη μορφή, να έχουμε την αίσθηση ότι τον βλέπουμε σε τρισδιάστατη μορφή. Αυτό γίνεται με την επίκληση της εντολής |3D για τη διαμόρφωση των εντολών PRINTSPALL.

και PRINTSP ώστε να προβάλλουν σε 3D πριν ζωγραφίσουν στην οθόνη. Δεν θα υπάρχει δυνατότητα περιστροφής στο επίπεδο, θα "κοιτάμε" πάντα προς τα εμπρός, αλλά το συνδυασμένο αποτέλεσμα μιας καμπύλης μαζί με σπίτια ή βουνά στον ορίζοντα που κινούνται, θα προσομοιώνει ότι παίρνουμε μια



καμπύλη.

Eik. 98 Τρισδιάστατη προβολή

- 2) **Zoom:** αυτό συνίσταται στη δυνατότητα χρήσης διαφορετικών εκδόσεων της ίδιας εικόνας ενός αντικειμένου για να δώσουμε ένα εφέ ζουμ καθώς το πλησιάζουμε. Αυτό γίνεται απλά στο αρχείο εικόνας, ορίζοντας 3 εκδόσεις της ίδιας εικόνας και ομαδοποιώντας τες σε μια δομή. Η εικόνα δείχνει το τυπικό παράδειγμα της αφίσας που κάνει ζουμ. Οι εντολές |PRINTSPALL και |PRINTSP θα επιλέξουν την καταλληλότερη έκδοση της εικόνας ανάλογα με την απόσταση στην οποία βρίσκεται, χωρίς να χρειάζεται να κάνετε κάτι περισσότερο από το να ορίσετε την εικόνα ως εικόνα τύπου "Zoom".



Eik. 99 Εικόνες ζουμ

- 3) **Segments:** αυτό συνίσταται στη δυνατότητα να έχουμε sprites τύπου "segment", που ορίζονται από μία μόνο οριζόντια γραμμή σάρωσης (ώστε να καταλαμβάνουν πολύ λίγο χώρο), στα οποία μπορούμε να συνδέσουμε ένα μήκος και μια κλίση. Με αυτά μπορούμε να κατασκευάσουμε δρόμους, ποτάμια κ.λπ. και πλαγιές δρόμων. Απλά θα το κάνουμε αυτό στο αρχείο εικόνας ορίζοντας έναν τύπο εικόνας που εκτός από πλάτος και ύψος έχει και κλίση. Θα χρησιμοποιήσουμε αυτά τα τμήματα στην κατασκευή του παγκόσμιου χάρτη.



Σχήμα 100 τμήματα με διαφορετικές κλίσεις

Τα τμήματα που χρησιμοποιούνται στο παιχνίδι "3D Racing one" είναι κόκκινα ή λευκά και παρόλο που είναι μεγάλα, ορίζονται μόνο από μια γραμμή σάρωσης. Για παράδειγμα, το αριστερό λευκό τμήμα αποτελείται από μερικά πράσινα bytes για το γρασίδι, μερικά λευκά και μερικά γκρι bytes για το δρόμο. Τη στιγμή της εκτύπωσης, το τμήμα που ορίζεται με αυτόν τον τρόπο αναπαράγεται στο επιθυμητό μήκος και εκτυπώνεται προοπτικά, έτσι ώστε ακόμη και αν πρόκειται για ευθύγραμμο τμήμα, θα φαίνεται στραβό.

Τέλος, αξίζει να αναφερθεί ότι σε πολλές περιπτώσεις θα χρειαστεί να ενημερώσετε δυναμικά τον χάρτη (εντολή **|UMAP**). Χάρη σε αυτή την εντολή, ο χάρτης μπορεί να είναι πολύ μεγάλος (κάτι που είναι απαραίτητο αν δημιουργήσουμε ένα κύκλωμα με πάρα πολλά τμήματα). Η εντολή αυτή περιγράφεται στο κεφάλαιο της κύλισης.
Στη συνέχεια, θα εξετάσουμε σε βάθος αυτές τις 3 λειτουργίες και πώς να τις χρησιμοποιήσετε στα προγράμματά σας.

18.1 3D προβολή

Για την προβολή έχουμε την εντολή **|3D**

Χρήση

|3D, 1, <sprite_fin>, <offsety> : REM αυτό ενεργοποιεί την τρισδιάστατη προβολή.

|3D, 0 REM απενεργοποιεί την τρισδιάστατη προβολή

Αυτή η εντολή ενεργοποιεί την τρισδιάστατη προβολή στην εντολή **|PRINTSP** και στην εντολή **|PRINTSPALL**. Αυτό σημαίνει ότι πριν από την εκτύπωση στην οθόνη, οι "προβαλλόμενες" συντεταγμένες θα υπολογιστούν και στη συνέχεια θα εκτυπωθούν στην οθόνη. Οι συντεταγμένες των sprites δεν επηρεάζονται, δηλαδή οι συντεταγμένες θα παραμείνουν οι ίδιες, αλλά στον 2D κόσμο. Στην οθόνη έχουμε τώρα μια τρισδιάστατη προβολή και οι συντεταγμένες όπου θα εκτυπωθούν θα είναι το αποτέλεσμα της εκτέλεσης της συνάρτησης προβολής στις 2D συντεταγμένες τους.

Τα sprites που επηρεάζονται είναι όλα από το Sprite 0 έως το **<sprite_fin>**. Τα υπόλοιπα sprites δεν θα προβάλλονται, απλώς θα εκτυπώνονται στην οθόνη στις 2D συντεταγμένες τους.

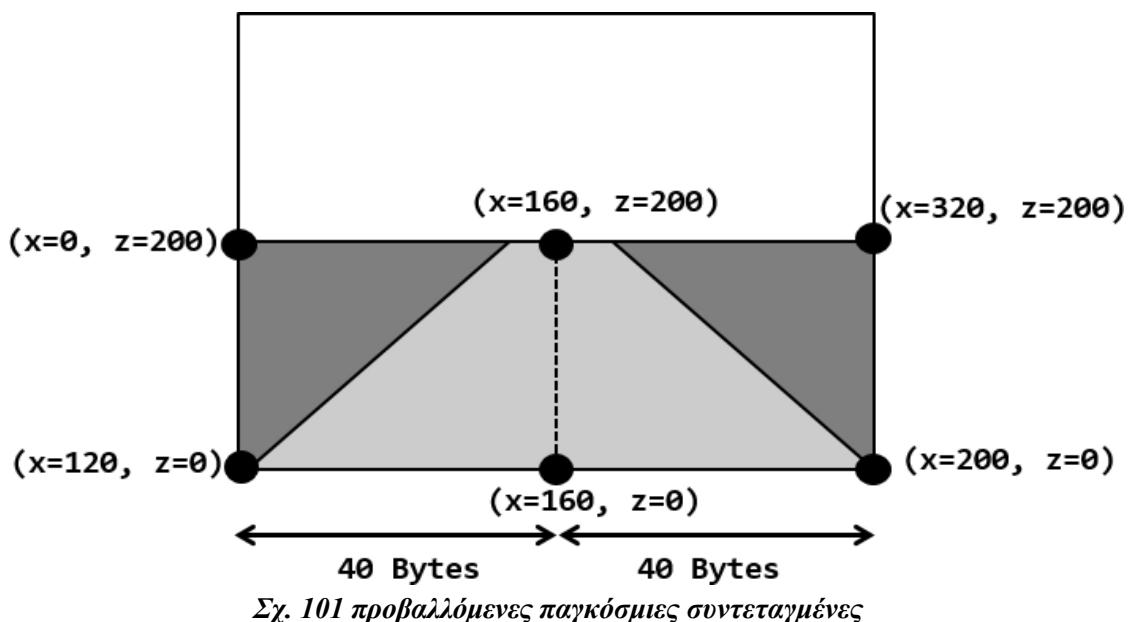
Αυτή η εντολή δεν επηρεάζει τους μηχανισμούς σύγκρουσης, δηλαδή αν

χρησιμοποιήσουμε COLSPALL και ανιχνεύσουμε μια σύγκρουση μεταξύ προβαλλόμενων sprites, η σύγκρουση θα συμβεί στο επίπεδο 2D. Μπορεί να συμβεί σε ορισμένες περιπτώσεις δύο προβαλλόμενα sprites να επικαλύπτονται μερικώς στο

οθόνη, αλλά αυτό δεν σημαίνει ότι έχουν συγκρουστεί, το ένα μπορεί να είναι ελαφρώς μπροστά από το άλλο και να επικαλύπτονται όταν προβάλλονται, αλλά αυτό δεν είναι πραγματική σύγκρουση. Οι συγκρουσεις ανιχνεύονται στο επίπεδο 2D.

Όσον αφορά την τελευταία παράμετρο `<offsety>` είναι να προβάλλουμε υψηλότερα ή χαμηλότερα, ώστε να μπορούμε να τοποθετήσουμε τους δείκτες του παιχνιδιού όπου θέλουμε. Κατά την προβολή της οθόνης, η οποία έχει ύψος 200 pixel, γίνεται 100 pixel ύψος, οπότε μπορούμε να επιλέξουμε πόσο ψηλά θα τοποθετήσουμε την προβολή. Αν ένα Sprite δεν επηρεάζεται από την προβολή επειδή είναι ψηλότερα από το `<Sprite_fin>`, τότε δεν επηρεάζεται ούτε από το `<offsety>`. Αυτή είναι η περίπτωση, για παράδειγμα, με τα σύννεφα και τα σπίτια στον ορίζοντα στο παιχνίδι "3D Racing one".

Για να καταλάβετε τις συντεταγμένες της οθόνης στις οποίες τελικά εμφανίζονται τα sprites που προβάλλετε, σας συνιστώ να διαβάσετε την παρακάτω πολύ απλή ενότητα μαθηματικών για να την κατανοήσετε πλήρως. Το παρακάτω σχήμα αναπαριστά ποιες είναι οι συντεταγμένες του παγκόσμιου χάρτη που προβάλλονται σε ορισμένα αντιπροσωπευτικά σημεία της οθόνης όταν το MAP2SP καλείται με ($yo=0$, $xo=0$). Η συντεταγμένη Z αντιπροσωπεύει την απόσταση στην οποία βρίσκονται τα αντικείμενα, που ονομάζεται επίσης "βάθος".



Αν αντί για ($xo=0$, $yo=0$) χρησιμοποιήσουμε άλλη συντεταγμένη για το MAP2SP, οι συντεταγμένες του κόσμου 2D που αντιστοιχούν στα σημεία που αναφέρονται στην εικόνα θα μετατοπιστούν κατά (x , z) όπως υποδεικνύεται από το (xo , yo).

18.1.1 Μαθηματικά της ψευδοτρισδιάστατης προβολής

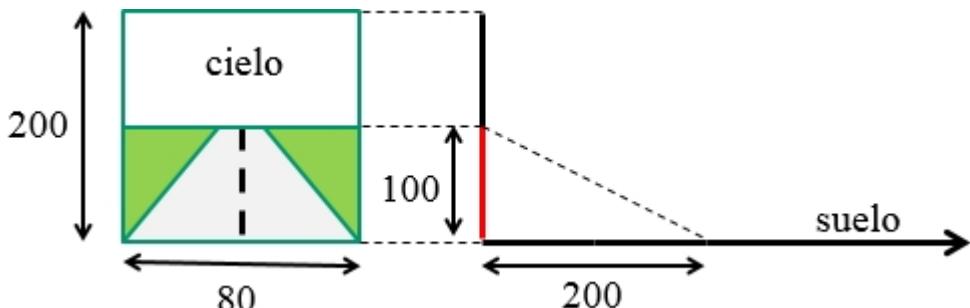
Η ψευδοτρισδιάστατη προβολή συνίσταται στην προβολή στην οθόνη του επιπέδου του εδάφους, όπου βρίσκεται ο δισδιάστατος χάρτης του κόσμου μας.

Υπολογισμός της συντεταγμένης Y

Το δάπεδό μας μπορεί να είναι άπειρο, αλλά θα προβάλλουμε μόνο 200 pixel. Αυτό το μήκος ονομάζεται "βάθος" ή συντεταγμένη "Z". Αυτά τα 200 εικονοστοιχεία βάθους,

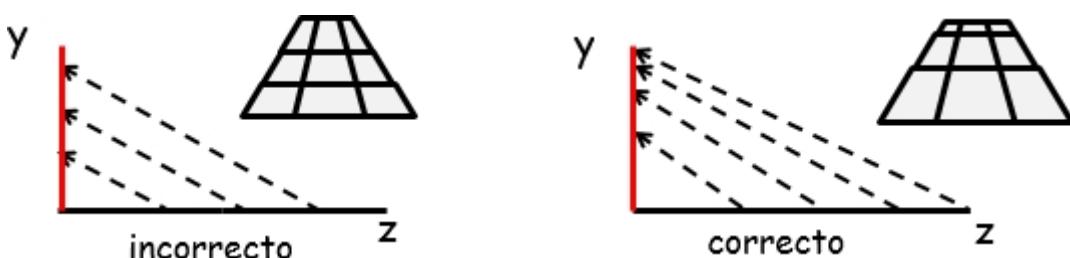
όταν προβάλλονται, συμπιέζονται σε 100 γραμμές ύψους (προβαλλόμενη συντεταγμένη Y). Τα εικονοστοιχεία

Τα πιο απομακρυσμένα προβαλλόμενα αντικείμενα είναι η γραμμή του ορίζοντα. Σημειώστε ότι δεν θα δούμε πολύ μακρινά αντικείμενα στον ορίζοντα, παρά μόνο αυτά που βρίσκονται σε βάθος 200 το πολύ.



Eik. 102 προβολή των δαπέδων στην οθόνη

Καθώς τα αντικείμενα απομακρύνονται, γίνονται όλο και μικρότερα. Δεν πρόκειται για γραμμική σχέση μεταξύ της οθόνης και του εδάφους, δηλαδή, για να ξέρουμε πόσο ψηλά πρέπει να εκτυπώσουμε ένα pixel που βρίσκεται σε ορισμένη απόσταση, δεν είναι σωστό να σκεφτούμε ότι καθώς το βάθος που καλύπτεται είναι 200 και προβάλλεται σε 100 γραμμές ύψος, αρκεί να το διαιρέσουμε με το δύο. Σε μια γραμμική συνάρτηση (όπως $y = f(z) = A \cdot z + B$) η παράγωγος (A) είναι σταθερή, αλλά στην προβολή, αυτό που είναι σταθερό είναι η "δεύτερη παράγωγος" της συνάρτησης $f(z)$. Αυτό θα το εξετάσουμε τώρα λεπτομερώς.



Σχ. 103 Η σωστή προβολή δεν είναι γραμμική

Ας υποθέσουμε ότι ξεκινάμε με " $Z=0$ " και θέλουμε να μάθουμε πόσο αξίζει το " Y ". Είναι πολύ εύκολο, στη γραμμή του εδάφους ($z=0$) η συντεταγμένη y είναι επίσης μηδέν.

Αν αυξήσουμε το z με μια μικρή μεταβολή " dz ", το " y " θα αξίζει το προηγούμενο " y " (μηδέν) συν μια αύξηση dy , η οποία αρχικά θα είναι ίση με την αύξηση dz , καθώς η αύξηση ήταν μικρή και βρισκόμαστε κοντά στον ορίζοντα.

$$dy (\text{αρχική}) = dz$$

Τώρα, αν απομακρυνθούμε και πάλι από το dz , η αύξηση dy πρέπει να μειωθεί, και αν απομακρυνθούμε και πάλι από το dz , το dy που πρέπει να προστεθεί θα είναι όλο και μικρότερο. Δηλαδή, το dy θα είναι όλο και μικρότερο:

Κάθε φορά που απομακρυνόμαστε από το dz , προσθέτουμε μια προσαύξηση στο y που κάθε φορά που απομακρυνόμαστε από το dz , προσθέτουμε μια προσαύξηση στο y που κάθε φορά που απομακρυνόμαστε από το dz ,

προσθέτουμε μια προσαύξηση στο y .

ο χρόνος είναι

μικρότερος $z=z+dz$

$dy = dy + ddy$, όπου το ddy είναι

αρνητικό $y= y + dy$

Η αύξηση του dy είναι σταθερή. Ονομάσαμε αυτή την αύξηση ddy. Λοιπόν, το dy είναι η "παράγωγος" του y, ενώ το "ddy" είναι αυτό που ονομάζεται "δεύτερη παράγωγος". Εδώ η παράγωγος δεν είναι σταθερή, αλλά η δεύτερη παράγωγος είναι.

Η σταθερή τιμή που αποδίδουμε στο "ddy" θα παράγει περισσότερο ή λιγότερο υπερβολικές προβολές. Στην 8BP η τιμή ddy είναι αρνητική, περίπου -0,005, καθιστώντας την dy στη γραμμή του εδάφους σχεδόν 1, ενώ στην οριζόντια γραμμή η συσσώρευση 200 ddy κάνει την τιμή dy να καταλήγει στο μηδέν.

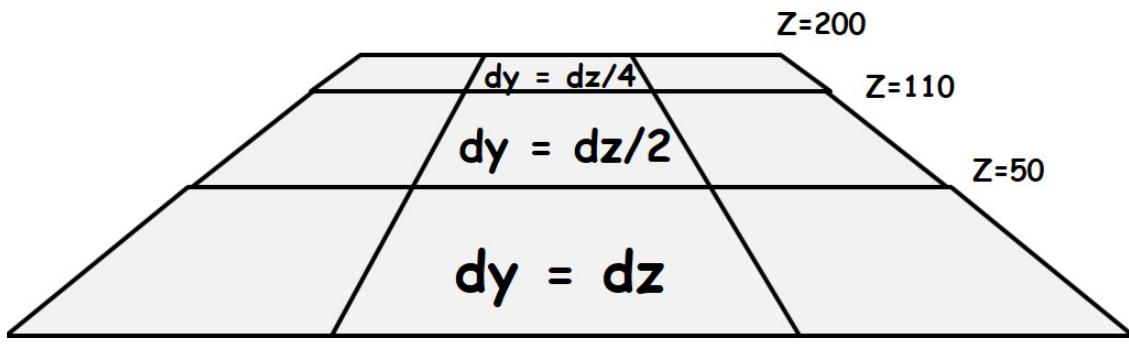
Παρά την απλότητα αυτών των υπολογισμών, η πραγματοποίησή τους σε πραγματικό χρόνο είναι δαπανηρή για το αγαπημένο μας Amstrad CPC, οπότε το 8BP κάνει στην πραγματικότητα μια προσέγγιση για να αποφύγει τους υπολογισμούς και να μεταφράσει το "z" σε "y" με πολύ απλούστερο τρόπο και με πολύ παρόμοιο αποτέλεσμα.

Εάν $0 \leq z < 50$, τότε $dy = dz$, επομένως $y = z$

Εάν $50 \leq z < 110$, τότε $dy = dz/2$, οπότε $y = 50 + (z-50)/2$

Εάν $110 \leq z \leq 200$, τότε $dy = dz/4$, οπότε $y = 50 + (110-50)/2 + (z-110)/2$

Δηλαδή, έχουμε χωρίσει την οθόνη σε τρεις λωρίδες και κάθε λωρίδα αντιμετωπίζεται ως "γραμμική" περιοχή (αφού το "dy" είναι σταθερό) αλλά με διαφορετική τιμή του "dy" σε σχέση με τις άλλες λωρίδες. Αυτή η προσέγγιση δίνει οπτικά πολύ παρόμοια αποτελέσματα με τα μαθηματικά ορθά.



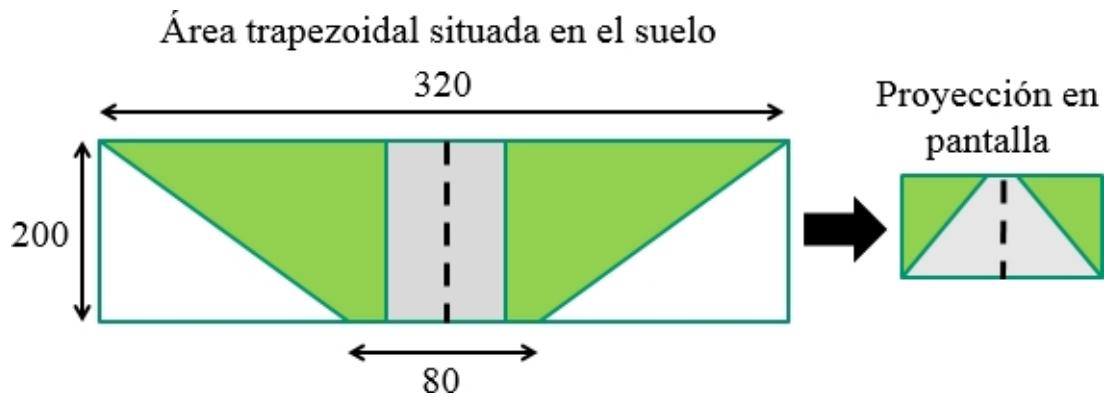
Σχ. 104 Προσέγγιση της 8BP

Οι εξισώσεις που χρησιμοποιούνται στο 8BP λαμβάνουν επίσης υπόψη ότι οι συντεταγμένες της οθόνης είναι στην πραγματικότητα "ανεστραμμένες", δηλαδή η συντεταγμένη μηδέν είναι η επάνω συντεταγμένη και η συντεταγμένη 200 είναι η κάτω συντεταγμένη, αλλά αυτό είναι απλώς μια πολύ απλή τελική προσαρμογή.

Ας περάσουμε τώρα στον υπολογισμό της συντεταγμένης "X":

Υπάρχει μια θεμελιώδης διαφορά μεταξύ της γραμμής του ορίζοντα και της γραμμής του εδάφους. Η γραμμή του εδάφους μετράει 80 bytes, αλλά η γραμμή του ορίζοντα αντιπροσωπεύει μεγαλύτερο πλάτος, επειδή ο δρόμος είναι ευθύς και αυτό που μετράει 80 στο έδαφος, μετράει ένα κλάσμα στην γραμμή του ορίζοντα, συγκεκριμένα στην 8BP μετράει 4 φορές λιγότερο. Ο δρόμος στενεύει στον ορίζοντα, επειδή ο ορίζοντας αντιπροσωπεύει πολύ περισσότερο. Στην προβολή που εφαρμόζει η 8BP αντιπροσωπεύει 320 bytes.

Και αν ο ορίζοντας μετράει 320 και το έδαφος 80, αυτό οφείλεται στο γεγονός ότι η συνολική πραγματική επιφάνεια που μπορούμε να δούμε στην οθόνη μετά την προβολή είναι τραπεζοειδής. ΔΕΝ είναι ότι το έδαφος είναι τραπεζοειδές, αλλά ότι η περιοχή του εδάφους που είμαστε σε θέση να δούμε στην οθόνη έχει τραπεζοειδές σχήμα.



Σχ. 105 Εμφανιζόμενη τραπεζοειδής περιοχή

Διαισθητικά και χωρίς να παρουσιάσουμε εξισώσεις, είναι ήδη σαφές τι θέλουμε να κάνουμε και πώς λειτουργεί η προβολή. Ας δούμε τώρα τις εξισώσεις.

Στην ουσία τα μαθηματικά της συντεταγμένης "X" είναι τα ίδια με εκείνα της συντεταγμένης "Y". Ωστόσο, δεν γίνονται με τον ίδιο τρόπο, διότι αφού έχουμε υπολογίσει τη συντεταγμένη "Y", μπορούμε να κάνουμε μια γραμμική εξίσωση $x=f(y)$, διότι λόγω της μη γραμμικότητας της "Y" σε σχέση με τη "Z", είναι σαν να δημιουργούμε μια μη γραμμική σχέση μεταξύ της "X" και της "Z". Νωρίτερα είπαμε ότι ο ορίζοντας έχει μήκος 320 bytes και το έδαφος έχει μήκος 80 bytes. Αυτό σημαίνει ότι το έδαφος είναι 4 φορές μικρότερο από τον ορίζοντα. Στην μακρινή απόσταση πρέπει να διαιρέσουμε με το 4 (η διαίρεση είναι ακριβή γι' αυτό προτιμούμε να πολλαπλασιάσουμε με συντελεστή 0,25) ενώ στην κοντινή απόσταση πρέπει να πολλαπλασιάσουμε με συντελεστή = 1.

Αυτό που πρέπει να κάνουμε είναι να κεντράρουμε τη συντεταγμένη X του αντικειμένου που πρόκειται να προβάλουμε σε σχέση με το κέντρο της οθόνης. Στη συνέχεια θα πολλαπλασιάσουμε με έναν παράγοντα που εξαρτάται από την προβαλλόμενη συντεταγμένη "Y". Με τον τρόπο αυτό θα δημιουργηθεί μια μη γραμμική σχέση μεταξύ "X" και "Z".

Πρέπει να κατασκευάσουμε μια εξίσωση που να επιστρέψει ένα αποτέλεσμα 4 φορές μικρότερο στον ορίζοντα, για παράδειγμα:

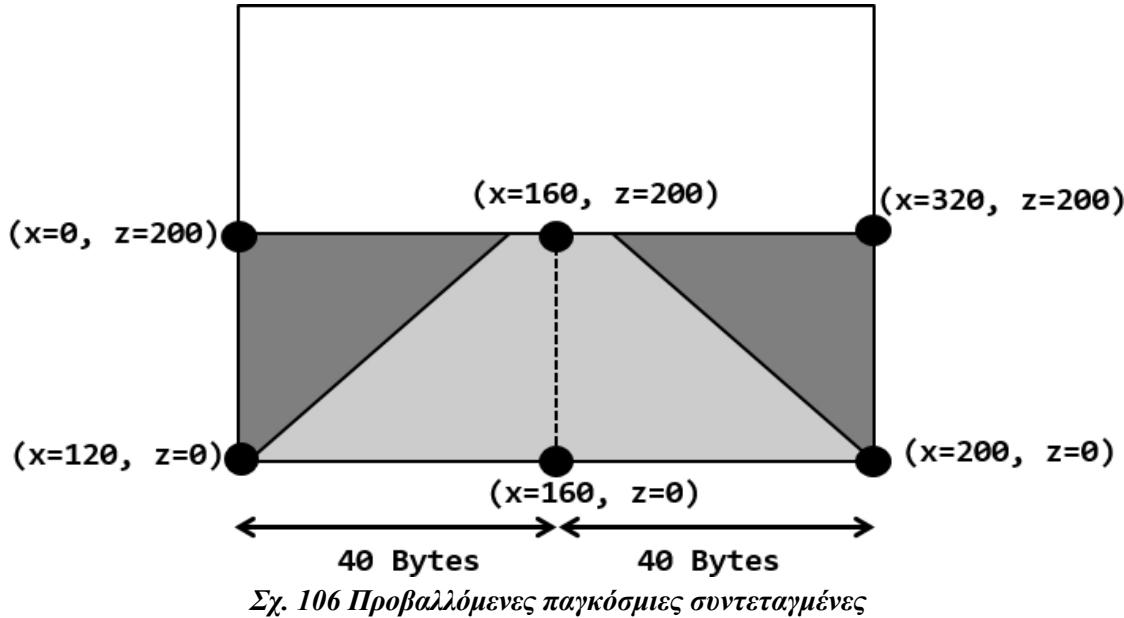
$\Sigma \text{ντελεστής} = ((100-y)+32)/2$ $x = (x - \text{κέντρο}) * \Sigma \text{ντελεστής} + \text{κέντρο}$ εάν $z=200$ τότε $y=100$, και στη συνέχεια παράγοντας =16 αν $z=0$ τότε $y=0$, και στη συνέχεια παράγοντας =64 (4 φορές περισσότερο)
--

Η επιλεγμένη εξίσωση είναι ενδιαφέρουσα επειδή η διαίρεση με το 2 είναι κάτι που το amstrad μπορεί να κάνει με την περιστροφή ενός bit στο δυαδικό σύστημα. Δηλαδή, μπορεί να το κάνει σε μερικούς κύκλους ρολογιού.

Όπως φαίνεται στην εξίσωση, για την προβολή της συντεταγμένης "X", αρκεί να την κεντράρετε και στη συνέχεια να την πολλαπλασιάσετε με τον συντελεστή που προκύπτει. Ο αριθμός που προκύπτει με αυτόν τον τρόπο είναι πολύ μεγάλος, διότι στην περίπτωση της γραμμής του ορίζοντα θα πολλαπλασιάσουμε με το 16 και στη γραμμή του εδάφους με το 64, δηλαδή έχουμε πολλαπλασιάσει τη συντεταγμένη X με έναν παράγοντα του οποίου η τιμή είναι μεταξύ 16 και 64. Μεταξύ του ορίζοντα και του εδάφους θα έχουμε και τις 48 πιθανές τιμές του συντελεστή, οπότε, αν και ο συντελεστής είναι ακέραιος, θα εξελίσσεται ομαλά.

Στη συνέχεια διαιρούμε με το 64, το οποίο είναι απλά η περιστροφή στο δυαδικό σύστημα 6 φορές, και αυτό είναι όλο. Το τελευταίο θα ισοδυναμεί με το να έχουμε πολλαπλασιάσει με 0,25 τον ορίζοντα, με 1 το έδαφος και με τη δεκαδική τιμή που αντιστοιχεί σε οποιοδήποτε ενδιάμεσο ύψος μεταξύ του εδάφους και του ορίζοντα, αλλά το έχουμε κάνει με ακέραιους αριθμούς, τους οποίους το Amstrad μπορεί να χειριστεί γρήγορα. Αυτή η τεχνική ονομάζεται "αριθμητική σταθερού σημείου".

Λαμβάνοντας όλα αυτά υπόψη και υποθέτοντας ότι ζητάμε από το MAP2SP να δημιουργήσει τα sprites του παγκόσμιου χάρτη από τις συντεταγμένες ($yo=0$, $xo=0$) αυτό που θα δούμε στην οθόνη θα έχει τις ακόλουθες συντεταγμένες του κόσμου. Είμαι σίγουρος ότι είναι εύκολο να καταλάβετε τώρα το σχήμα.



Όπως μπορείτε να συμπεράνετε, το σημείο ($x=0, y=0$) του παγκόσμιου χάρτη προβάλλεται εκτός της οθόνης και δεν εμφανίζεται. Αν αντί για ($xo=0, yo=0$) χρησιμοποιήσουμε μια άλλη συντεταγμένη για το MAP2SP, οι 2D παγκόσμιες συντεταγμένες που αντιστοιχούν στα σημεία που αναφέρονται στην εικόνα θα μετατοπιστούν κατά (x, z) όπως υποδεικνύεται από το (xo, yo).

18.1.2 Καμπύλες

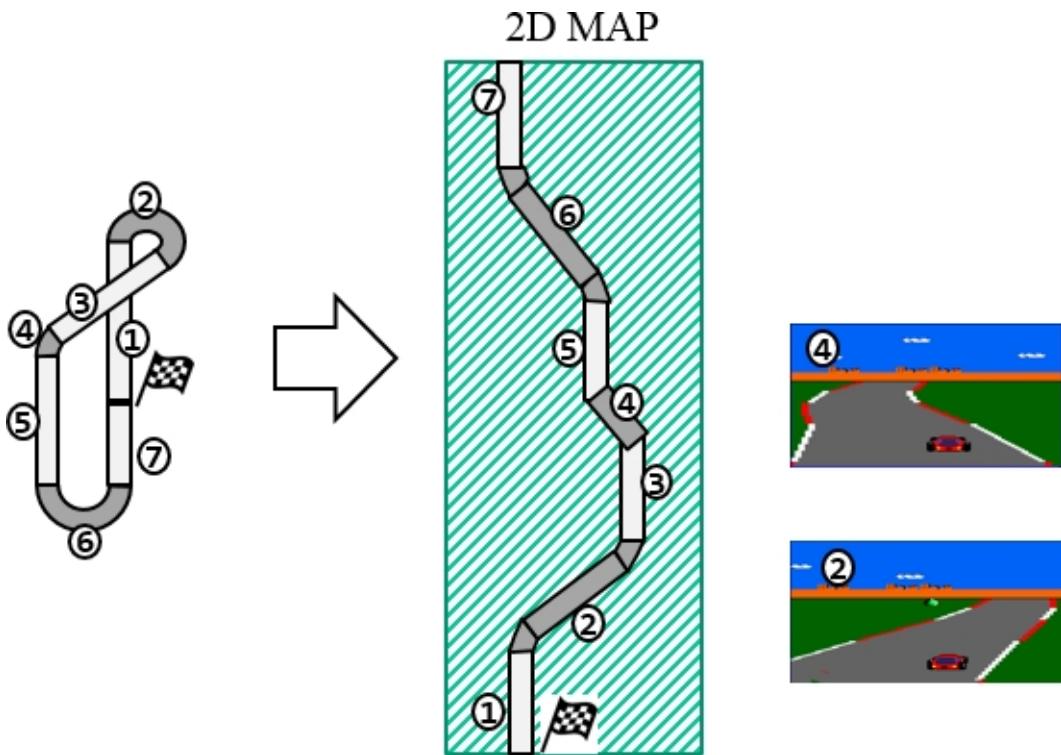
Όπως ανέφερα στην αρχή αυτού του κεφαλαίου, η τρισδιάστατη προβολή που χρησιμοποιεί το 8BP δεν επιτρέπει περιστροφές του επιπέδου, οπότε για να προσομοιώσουμε μια καμπύλη θα πρέπει να κάνουμε ένα μικρό κόλπο. Αυτό περιλαμβάνει τη στροφή του δρόμου προς τα δεξιά ή προς τα αριστερά, ενώ ταυτόχρονα μετακινούμε sprites όπως σπίτια ή βουνά στον ορίζοντα. Αυτά τα sprites δεν θα προβάλλονται και για να το αποφύγουμε αυτό θα χρησιμοποιήσουμε αναγνωριστικά πάνω από το `<Sprite_fin>`. Θυμηθείτε ότι για να ενεργοποιήσουμε την τρισδιάστατη προβολή θα κάνουμε:

| 3D, 1, <Sprite_fin>, <offsety>, <offsety>, <offsety>, <offsety>, <offsety>, <offsety>.

Κατά συνέπεια, ένα προφανές κύκλωμα με καμπύλες θα αναπαρασταθεί στον δισδιάστατο χάρτη μας ως ένας δρόμος με κλίσεις προς τα δεξιά και προς τα αριστερά, απλά.

Με αυτή τη στρατηγική, μπορούμε να δημιουργήσουμε την αίσθηση ενός οδηγού αγώνων που οδηγεί σε μια πίστα με πραγματικές καμπύλες και να δώσουμε την εντύπωση ότι το αυτοκίνητό του στρίβει στις καμπύλες μέσω σπιτιών ή βουνών στον ορίζοντα που κινούνται προς την αντίθετη κατεύθυνση από τη συντεταγμένη X. Αν είστε καλός καλλιτέχνης και στρίβετε σταδιακά διαφορετικά τμήματα σε μεγαλύτερο ή μικρότερο βαθμό, δίνετε την εντύπωση πολύ ρεαλιστικών καμπυλών.

Αυτή η στρατηγική είναι υπολογιστικά πολύ αποδοτική και αρκετά ρεαλιστική. Αν θέλαμε να περιστρέψουμε το επίπεδο του εδάφους πραγματικά, θα έπρεπε να εφαρμόσουμε υπολογισμό πινάκων για την περιστροφή, με πολλές πολύ ακριβές πράξεις και, επιπλέον, οι υφές των sprites στο έδαφος θα έπρεπε επίσης να περιστραφούν, οπότε το υπολογιστικό κόστος θα ήταν τεράστιο.



Σχ. 107 φανταστικό κύκλωμα και 2D παιγκόσμιος χάρτης

Οι σημερινοί υπολογιστές είναι τόσο ισχυροί που οι στρατηγικές που παρουσιάζονται εδώ δεν έχουν νόημα, αλλά είναι στρατηγικές ανώτερες σε κομψότητα και εφευρετικότητα από τη σημερινή "ωμή βία". Να θυμάστε ότι "οι περιορισμοί δεν αποτελούν πρόβλημα, αλλά πηγή έμπνευσης".

Θα χρειαστεί να χρησιμοποιήσετε την εντολή **|UMAP** για να διασχίσετε μεγάλους χάρτες κυκλωμάτων, αλλά να θυμάστε ότι πρέπει να καλείτε την εντολή **|UMAP** όχι σε κάθε κύκλο αλλά μόνο κατά διαστήματα.

18.2 Μεγέθυνση εικόνων

Για να ορίσουμε μια "εικόνα ζονμ", απλά δημιουργούμε τις διαφορετικές εκδόσεις της εικόνας (3 εκδόσεις). Στη συνέχεια, στο αρχείο εικόνας "images_mygame.asm" θα αναζητήσουμε μια ετικέτα που ονομάζεται "**_3D_ZOOM_IMAGES**".

Θα το βρούμε αυτό:

```
_BEGIN_3D_ZOOM_IMAGES
;=====
;όρια που ισχύουν για όλες τις εικόνες με μεγέθυνση
Ο ορίζοντας για τα όρια αυτά θεωρείται ότι είναι το 0 και αυξάνεται προς τα κάτω έως το 200
_ZOOM_LIMIT_A
db 120- μεταξύ 200 (έδαφος) και το limitA έχει οριστεί στην εικόνα 3
_ZOOM_LIMIT_B
db 50
μεταξύ αυτού του ορίου και του ορίου A, ορίζεται η εικόνα 2.
πιο κοντά στον ορίζοντα από το όριο B τίθεται εικόνα 1
;=====
```

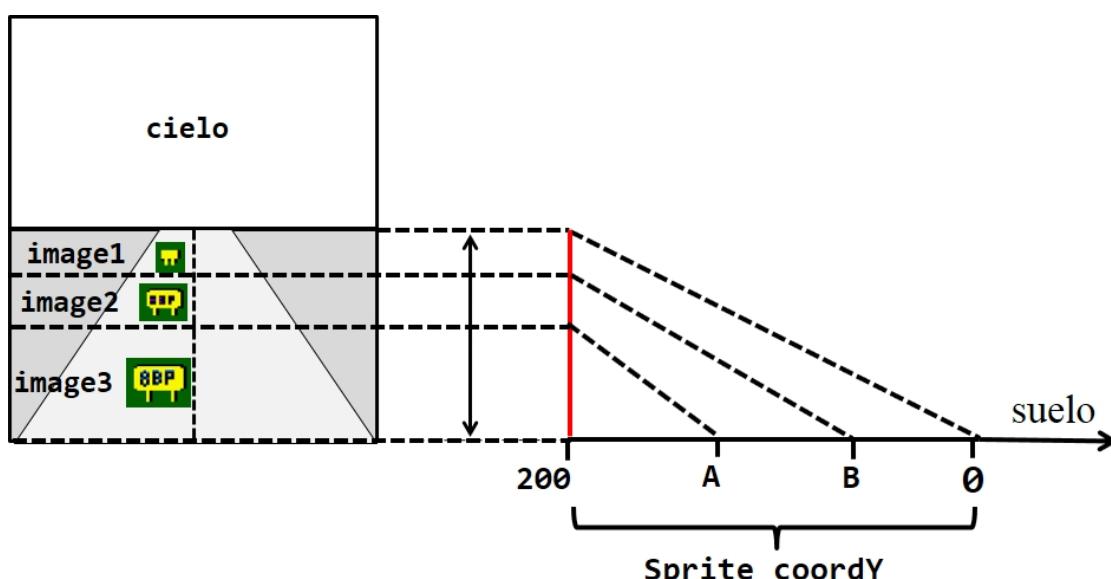
CARTEL_ZOOM**db 1, πλάτος συμβολικό****db 1, ύψος συμβολικό****dw POSTER1, POSTER2, POSTER3****_END_3D_ZOOM_IMAGES**

Όλες οι εικόνες ZOOM πρέπει να ορίζονται μετά την ετικέτα "BEGIN_3D_ZOOM_IMAGES". Πρόκειται για εικόνες που έχουν ένα συμβολικό πλάτος και ύψος, διότι στην πραγματικότητα ένα Sprite στο οποίο ανατίθεται μία από αυτές τις εικόνες, θα χρησιμοποιεί το πλάτος και το ύψος της έκδοσης της εικόνας που επιλέγεται αυτόματα σύμφωνα με την συντεταγμένη Y του. Δηλαδή, το "CARTEL1" είναι μια κανονική εικόνα, που έχει οριστεί προηγουμένως, με το πλάτος, το ύψος και τα bytes που περιέχει το σχέδιο. Το ίδιο ισχύει και για τις εικόνες "CARTEL2" και "CARTEL3". Αν συσχετίσουμε την εικόνα "CARTEL_ZOOM" με ένα Sprite (αντό μπορεί να γίνει συσχετίζοντας ένα αναγνωριστικό σε αυτή την εικόνα στην αρχή του αρχείου εικόνας) αυτό που θα συμβεί είναι ότι, ανάλογα με τη θέση του Sprite στην οθόνη, θα εκτυπωθεί η μία ή η άλλη εκδοχή της εικόνας.

Για την αυτόματη επιλογή της εικόνας, ορίζονται κατώφλια συντεταγμένων y. Μπορείτε να αλλάξετε αυτά τα κατώφλια, αλλά τα προεπιλεγμένα λειτουργούν μια χαρά και είναι τα εξής:

- μεταξύ ορίζοντα =0 και 50, επιλέγεται η πρώτη εικόνα (στο παράδειγμα, "CARTEL1").
- μεταξύ 50 και 120, επιλέγεται η δεύτερη εικόνα (στο παράδειγμα, "CARTEL2").
- μεταξύ 120 και 200, επιλέγεται η τρίτη εικόνα (στο παράδειγμα, "CARTEL3").

Η επιλογή αυτών των ορίων για την οριοθέτηση των ορίων της οθόνης είναι παραμετροποιήσιμη και μπορείτε να ορίσετε όσα όρια θέλετε. Σημειώστε ότι η επιλογή γίνεται με βάση τη συντεταγμένη Y του μη προβαλλόμενου Sprite. Μόλις προβάλλεται, η συντεταγμένη Y του μεταβάλλεται πολύ, αλλά δεν είναι η προβαλλόμενη "Y" που χρησιμοποιείται για την οριοθέτηση των 3 λωρίδων, αλλά η "Y" του Sprite σε 2D. Όταν το Sprite μετακινείται από τη μία λωρίδα στην άλλη, η εικόνα του θα αλλάζει αυτόματα. Αν προτιμάτε να εμφανίζεται πρώτα μια εικόνα, μπορείτε να τροποποιήσετε τα κατώτατα όρια.



Σχ. 108 Εύρος χρήσης των 3 εκδόσεων της εικόνας ZOOM

Για παράδειγμα:

```
REM ας υποθέσουμε ότι το CARTEL_ZOOM έχει id=16 στο αρχείο εικόνας  
|SETUPSP,20,9,16 : REM συσχετίζει το CARTEL_ZOOM με το Sprite 20  
|LOCATESP, 20,100, 160: REM Sprite στο at κέντρο του  
    "τραπεζοειδές  
(coordy=100)  
|3D,1,31,0: REM όλα τα sprites θα προβάλλονται.  
|PRINTSP,20: REM Εκτυπώνεται η έκδοση 2 του POSTER.
```

18.3 Χρήση τμημάτων

Τώρα που ξέρετε πώς να κατασκευάσετε έναν 2D χάρτη που "προσομοιώνει" μια πίστα με καμπύλες, εδώ είναι ένας ισχυρός τρόπος για να κατασκευάσετε τα τμήματα με διαφορετικούς βαθμούς κλίσης που χρειάζεστε για να κατασκευάσετε αγωνιστικές πίστες ή δρόμους.

Ένα τμήμα είναι μια ενιαία γραμμή υψηλής εικόνας, η οποία με την επανάληψη μπορεί να φτάσει σε οποιοδήποτε μήκος θέλουμε. Εκτός από το μήκος, έχει και μια άλλη παράμετρο, η οποία είναι η συνολική κλίση του τμήματος, σε bytes. Αυτή η κλίση μπορεί να είναι αρνητική (κλίση προς τα δεξιά) ή θετική (κλίση προς τα αριστερά).

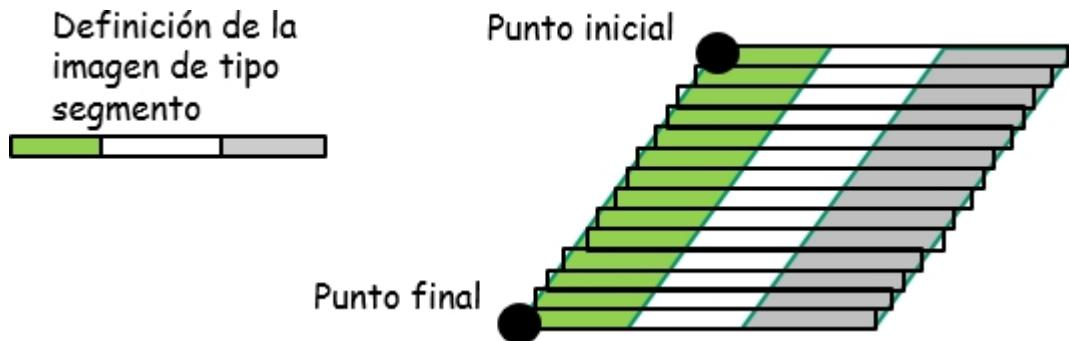
Για να ορίσετε αυτόν τον τύπο εικόνων πρέπει να τις δημιουργήσετε στο αρχείο εικόνων του παιχνιδιού σας "images_mygame.asm", μετά την ετικέτα "_BEGIN_3D_SEGMENTS".

```
;=====  
_BEGIN_3D_SEGMENTS  
;=====  
Θα μπορούσε να υπάρχει διαφορετικός ορισμός των τμημάτων.  
; το πλάτος είναι το πλάτος της γραμμής σάρωσης  
; το υψηλό είναι το 2D υψηλό του τμήματος  
Στη συνέχεια, ακολουθεί το dx, το οποίο μπορεί να είναι θετικό (κλίση προς τα αριστερά) ή αρνητικό (κλίση προς τα δεξιά).  
db 0; αυτό είναι για την πρώτη εικόνα τύπου τμήματος που θα είναι >  
_3D_SEGMENTS  
  
;----- SEGMENT_EDGE_LWI10 -----  
SEGMENT_EDGE_LWI10  
db 22; πλάτος  
db 50; υψηλή  
db 10; dx  
db  
192,192,192,192,192,192,192,192,192,192,192,192,192,192,192,192,  
240, 240  
,0,0,0,0,0,0,0,0,0  
;
```

Στο παράδειγμα έχουμε ορίσει ένα τμήμα που έχει γρασίδι στα αριστερά (πράσινα bytes), στη συνέχεια δύο λευκά bytes και στη συνέχεια γκρι bytes (δρόμος) στα δεξιά. Το αν είναι πράσινο ή γκρι εξαρτάται από τα χρώματα που έχουμε συσχετίσει με τα μελάνια.

Πρόκειται για ένα τμήμα με κλίση προς τα αριστερά 10 bytes. Αν είχαμε βάλει ένα μηδέν στην κλίση (dx), θα ήταν ένα ευθύγραμμο τμήμα, όχι λοξό. Έχει ύψος 50 γραμμές, αλλά όταν προβάλλεται είναι μικρότερο, εκτός αν είναι πολύ κοντά.

Τα σημεία του τμήματος που προβάλλονται είναι μόνο δύο. Αφού προβάλλονται, οι γραμμές σάρωσης ζωγραφίζονται μία προς μία με ορισμένη οριζόντια μετατόπιση, έτσι ώστε η τελευταία γραμμή να καταλήγει να ξεκινά από το τελικό σημείο. Σημειώστε ότι, παρόλο που το τμήμα ζωγραφίζεται προοπτικά, το πλάτος του είναι σταθερό. Δεν ζωγραφίζετε το πάνω τμήμα στενότερο (πιο μακριά) και το κάτω τμήμα πλατύτερο (πιο κοντά), αλλά ζωγραφίζετε κάθε γραμμή σάρωσης ακριβώς όπως έχει οριστεί στο αρχείο εικόνας.



Σχ. 109 2 σημεία προβάλλονται σε ένα τμήμα

Όπως μπορείτε να δείτε, έχω χρησιμοποιήσει πολύ γρασίδι και bytes δρόμου. Αυτό γίνεται έτσι ώστε το τμήμα να "σβήνει μόνο του" καθώς κινείται προς τα εμπρός, αν και αν το αυτοκίνητο πηγαίνει πολύ γρήγορα, μπορεί να παραμείνουν ίχνη. Μόλις βάλετε το παιχνίδι σας σε λειτουργία θα ελέγξετε αν η ταχύτητα είναι πολύ γρήγορη και αν έχουν μείνει ίχνη.



Σχ. 110 Σχέση των βαθμού κλίσης ενός τμήματος και της μέγιστης ταχύτητας

Όπως φαίνεται στο σχήμα, ο μέγιστος ρυθμός πρόσωσης για ένα αυτοδιαλυόμενο τμήμα μπορεί να είναι υψηλότερος εάν ο βαθμός στρέβλωσης είναι μέτριος. Εάν είναι πολύ στραβό, η ταχύτητα δεν μπορεί να είναι τόσο μεγάλη, αλλιώς θα παραμείνουν ίχνη. Μόλις το τμήμα προβάλλεται, θα είναι μικρότερο λόγω του φαινομένου της προοπτικής, και ανάλογα με το πού βρίσκεται, μπορεί να είναι ακόμη περισσότερο ή λιγότερο στραβό. Καλό είναι να τα κάνετε πλατιά, ώστε να σβήνουν μόνα τους με μεγαλύτερη ασφάλεια.

Στο παιχνίδι **"3D Racing one"** υπάρχει ένα στάδιο που ονομάζεται **"superfast"**, στο οποίο, χρησιμοποιώντας λιγότερο λυγισμένα τμήματα, αύξησα την ταχύτητα του αυτοκινήτου χωρίς τα τμήματα να αφήνουν ίχνη. Ένα απλό και πολύ αποτελεσματικό "κόλπο". Αν το παιχνίδι είναι αργό (π.χ. ένα παιχνίδι με τανκς, το οποίο υποτίθεται ότι είναι αργό), τότε τα τμήματα μπορούν να είναι πολύ στραβά, καθώς δεν θα αφήσουν ίχνη λόγω του φαινομένου της ταχύτητας.

Εν ολίγοις, για να αυξήσετε την ταχύτητα έχετε δύο επιλογές:

- Χρησιμοποιήστε λιγότερο στριμμένα τμήματα
- Χρησιμοποιήστε ευρύτερα τμήματα (με μεγαλύτερο περιθώριο για να διαγραφούν).

19 Μουσική

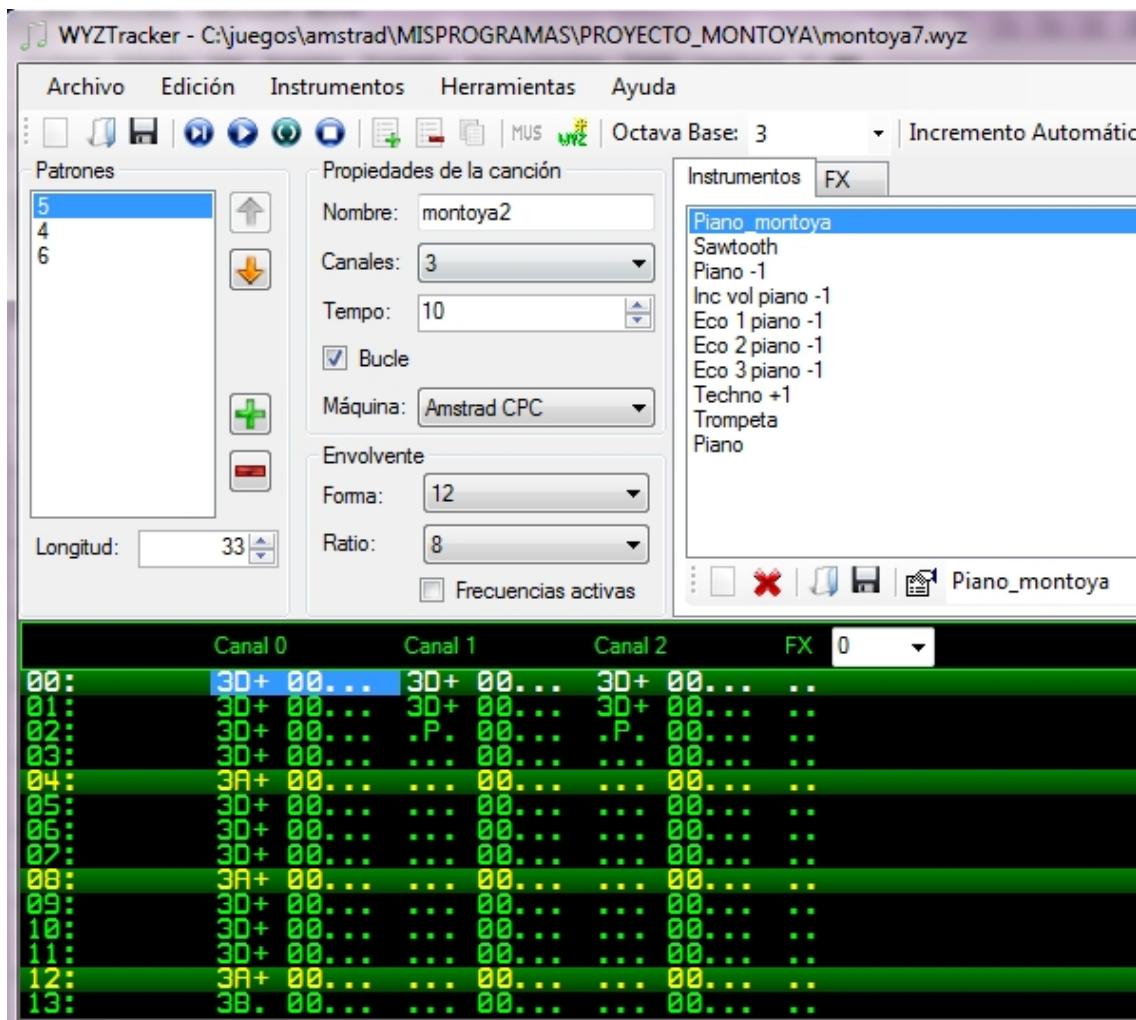
Τα εργαλεία για τα οποία θα μιλήσω σε αυτή την ενότητα δεν έχουν προγραμματιστεί από εμένα, αλλά είναι ενσωματωμένα στο 8BP και είναι πραγματικά καλά.

19.1 Επεξεργασία μουσικής με τον ιχνηλάτη Wyz

Αυτό το εργαλείο είναι ένα μουσικό sequencer για το τσιπ ήχου AY3-8912. Η μουσική που παράγει μπορεί να εξαχθεί και τα αποτελέσματα σε δύο αρχεία

- Ένα αρχείο οργάνου ".mus.asm".
- Ένα αρχείο με μουσικές νότες ".mus".

Μπορείτε να συνθέσετε τραγούδια με αυτό το εργαλείο και ο μόνος περιορισμός είναι ότι όλα τα τραγούδια που ενσωματώνετε στο παιχνίδι σας πρέπει να έχουν το ίδιο αρχείο οργάνων.



Εικ. 111 WYZTracker

Από την έκδοση V41 μπορείτε να χρησιμοποιήσετε το κανάλι "FX" για να ενσωματώσετε ηχητικά εφέ. Μέχρι αυτή την έκδοση αυτό το ψευδοκανάλι αγνοοούνταν στο πρόγραμμα αναπαραγωγής 8BP. Να είστε προσεκτικοί όταν επεξεργάζεστε τη μουσική επειδή, παρόλο που το κανάλι FX συνδέεται πάντα με ένα κανάλι, μέσα στο WYZ tracker ακούγεται σαν να ήταν ανεξάρτητο και το αποτέλεσμα μπορεί να είναι

διαφορετικό από όταν το ακούτε στο WYZ tracker.

Εάν το FX συσχετιστεί με το κανάλι 0 και τοποθετήσετε ένα FX τη στιγμή που το κανάλι 0 δεν παίζει, δεν θα ακουστεί τίποτα στο amstrad ακόμα και αν το εφέ σας ακούγεται στο WYZtracker.

Αυτό το μουσικό sequencer συμπληρώνεται από το WYZplayer που είναι ενσωματωμένο στη βιβλιοθήκη 8BP.

Από την έκδοση V26 του 8BP, η μουσική μπορεί να συντίθεται με την έκδοση 2.0.1.0 του WYZtracker και λειτουργεί πολύ καλά. Μέχρι την έκδοση V25 του 8BP η συμβατότητα ήταν με το WYZtracker 0.5.07 και υπήρχαν κάποια μικρά προβλήματα, αλλά όλα αυτά εξαφανίστηκαν με το WYZtracker 2.0.1.0.

ΣΗΜΑΝΤΙΚΟ: μην χρησιμοποιείτε την οκτάβα 7 ακόμη και αν ο ιχνηλάτης σας επιτρέπει να τη χρησιμοποιήσετε. Αυτή η οκτάβα δεν μπορεί να αναπαραχθεί και ενδέχεται να προκύψουν σφάλματα συγχρονισμού κατά την αναπαραγωγή μουσικής στο Amstrad.

Μια σημαντική σύσταση κατά τη δημιουργία των τραγουδιών σας με το WyZtracker είναι να διαγράφετε τα όργανα που δεν πρόκειται να χρησιμοποιήσετε. Με αυτόν τον τρόπο το αρχείο οργάνων (το οποίο τελειώνει σε ".mus.asm") θα καταλαμβάνει λιγότερο χώρο, και δεδομένου ότι το 8BP διατηρεί μόνο **1.500 bytes** για τη μουσική, κάθε byte είναι σημαντικό.

ΣΗΜΑΝΤΙΚΟ: αν επεξεργαστείτε τη μουσική με μια έκδοση του WYZtracker μεταγενέστερη της έκδοσης 2.0.1.0 ενδέχεται να μην λειτουργεί σωστά και να αντιμετωπίσετε περιέργα φαινόμενα, όπως η μη αναπαραγωγή ενός καναλιού ή η μη συγχρονισμός των νοτών. Αν σας συμβεί αυτό, η λύση είναι να δημιουργήσετε ένα αρχείο από την αρχή με το WYZtracker 2.0.1.0 και να αντιγράψετε χειροκίνητα νότα προς νότα τη μουσική που δεν λειτουργεί.

19.2 Συναρμολόγηση των τραγουδιών

Μόλις συνθέσετε το τραγούδι σας και έχετε τα δύο αρχεία, απλά επεξεργάξεστε το αρχείο make_music.asm και συμπεριλαμβάνετε τα αρχεία μουσικής σας ως εξής:

**μετά τη συναρμολόγηση, αποθηκεύστε το με save
"musica.bin",b,32200,1400**

org 32200

;-----MUSICA-----
Ο περιορισμός είναι ότι μπορείτε να συμπεριλάβετε μόνο ένα αρχείο του
; όργανα για όλα τα τραγούδια
Ο περιορισμός επιλύεται με την απλή τοποθέτηση όλων των όργανα σε ένα ενιαίο αρχείο.

**αρχείο οργάνων. ΣΗΜΕΙΩΣΗ ΠΡΕΠΕΙ ΝΑ ΕΙΝΑΙ
MONO MIA ανάγνωση "instruments.mus.asm"**

**; αρχεία μουσικής SONG_0:
INCBIN "micancion.mus"; 217
SONG_0_END:**

SONG_1:

```

INCBIN      "another_song.mus"
; SONG_1_END:

SONG_2:
INCBIN      "third_song.mus"      ;
SONG_2_END:
SONG_3:
SONG_4:
SONG_5:
SONG_6:
SONG_7:

```

Τέλος, επανασυναρμολογείτε τη βιβλιοθήκη 8BP έτσι ώστε η συσκευή αναπαραγωγής μουσικής (η οποία είναι ενσωματωμένη στη βιβλιοθήκη) να γνωρίζει τις παραμέτρους των οργάνων και το μέρος όπου έχουν συγκεντρωθεί τα τραγούδια.

Για να το κάνετε αυτό, απλά συναρμολογείτε το αρχείο make_all.asm, το οποίο μοιάζει με αυτό

```

; Makefile για βιντεοπαιχνίδια που χρησιμοποιούν ισχύ 8bit
αν αλλάξετε μόνο ένα μέρος, θα πρέπει να συναρμολογήσετε μόνο το
αντίστοιχο make
για παράδειγμα, μπορείτε να συναρμολογήσετε το make_graphics αν
αλλάξετε σχέδια
-----CODIGO -----
;περιλαμβάνει τη βιβλιοθήκη 8bp και το music
playerWYZ διαβάστε το "make_codigo.asm".

-----MUSICA-----
;περιλαμβάνει τα
τραγούδια. διαβάστε
"make_musica.asm"

-----ΓΡΑΦΗΜΑΤΑ -----
Αντό το μέρος περιλαμβάνει εικόνες και ακολουθίες κινουμένων
σχεδίων.
και ο πίνακας sprite αρχικοποιείται με αυτές τις εικόνες και τις
ακολουθίες που διαβάζονται στο "make_graphics.asm".

```

Και με αυτό έχετε συναρμολογήσει τα πάντα. Τώρα πρέπει να δημιουργήσετε τη βιβλιοθήκη 8BP ως εξής:

ΑΠΟΘΗΚΕΥΣΗ "8BP.LIB", b, 24000, 8200

Και η μουσική:

ΑΠΟΘΗΚΕΥΣΗ "music.bin", b, 32200, 1400

19.3 Τι να κάνετε αν δεν μπορείτε να χωρέσετε μουσική σε 1400 bytes

Είναι πιθανό τα 1400 bytes να μην είναι αρκετά για τα τραγούδια σας. Σε περίπτωση που ένα τραγούδι δεν χωράει (και θα το καταλάβετε αυτό ελέγχοντας πού

συναρμολογείται η ετικέτα "_END_MUSIC") μπορείτε να καθορίσετε μια διαφορετική διεύθυνση συναρμολόγησης για το τραγούδι αυτό και τα επόμενα τραγούδια. Στην περίπτωση του βιντεοπαιχνιδιού "Nibiru", το κάνετε αυτό με το τρίτο τραγούδι, συναρμολογώντας το σε χαμηλότερη διεύθυνση από τη βιβλιοθήκη 8BP (π.χ. η 23000 θα λειτουργούσε). Σε περίπτωση που το κάνετε αυτό, το παιχνίδι BASIC σας θα πρέπει να ξεκινήσει με ένα

Η εντολή MEMORY που προσδιορίζει μια διεύθυνση χαμηλότερη από αυτό το νέο όριο, π.χ. MEMORY 22999, θα λειτουργούσε.

Από την έκδοση 34 της 8BP και μετά, η βιβλιοθήκη συναρμολογείται από τη διεύθυνση 24000, οπότε αν θέλετε να χρησιμοποιήσετε επιπλέον χώρο για μουσική, πρέπει να καταλαμβάνει διευθύνσεις χαμηλότερες από την 24000. Για παράδειγμα, από 23500 έως 23999.

Αυτό είναι το

**μετά τη συναρμολόγηση, αποθηκεύστε το με save
"musica.bin",b,32200,1400**

org 32200

;-----MUSICA-----

**περιορίζεται στο ότι μπορεί να συμπεριλάβει μόνο ένα αρχείο
οργάνων για την**

**Ο περιορισμός λύνεται με την απλή εισαγωγή όλων των τραγουδιών.
όργανα σε ένα ενιαίο αρχείο.**

**αρχείο οργάνων. ΣΗΜΕΙΩΣΗ ΠΡΕΠΕΙ ΝΑ ΕΙΝΑΙ
MONO MIA ανάγνωση ".../MUSIC/nibiru5.mus.asm"**

**; αρχεία μουσικής SONG_0:
INCBIN ".../MUSIC/attack5.mus" ;
SONG_0_END:**

**SONG_1:
INCBIN ".../MUSIC/nibiru5.mus" ;
SONG_1_END:**

**org 23500 ; Χρησιμοποιώ αυτή τη γραμμή επειδή δεν μπορώ να χωρέσω το
τρίτο τραγούδι !!!**

**SONG_2:
INCBIN ".../MUSIC/gorgo3.mus" ,**

**SONG_3:
SONG_4:
SONG_5:
SONG_6:
SONG_7:
_END_MUSIC**

Ο ίδιος τύπος λύσης ισχύει και στην περίπτωση που δεν μπορείτε να χωρέσετε όλα τα γραφικά σας στην περιοχή που δεσμεύεται από το 8BP, αν και καθώς έχετε 8540 bytes για τα γραφικά είναι λιγότερο πιθανό να αντιμετωπίσετε αυτό το πρόβλημα.

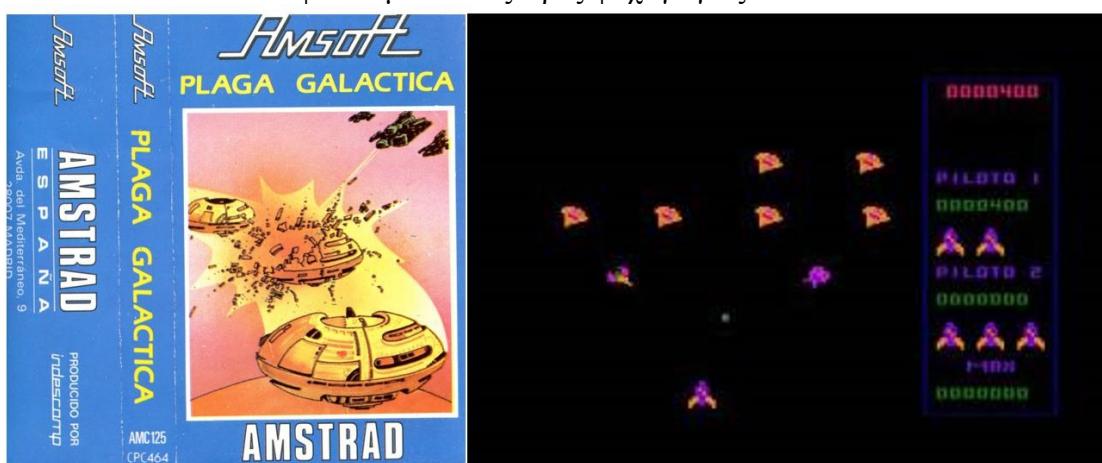
20 Προγραμματισμός σε C με 8BP

Στην αρχή αυτού του εγχειριδίου σας συνέστησα να μην χρησιμοποιείτε μεταγλωττιστές BASIC όπως το Fabacom ή το CPC BASIC 3 λόγω των περιορισμών μνήμης που επιβάλλουν (χάνετε περίπου 20KB). Αν αυτό που θέλετε είναι να αυξήσετε την ταχύτητα των παιχνιδιών σας, από το 8BP v40 και μετά έχετε στη διάθεσή σας ένα περιτύλιγμα της βιβλιοθήκης 8BP που μπορεί να χρησιμοποιηθεί από τη C και ένα μικρό σύνολο εντολών BASIC που μπορούν να κληθούν από τη C (το οποίο ονομάζω "minibasic"), ώστε να μπορείτε να μεταφράσετε το πρόγραμμα BASIC σας σχεδόν αμέσως και να έχετε το γρήγορο αποτέλεσμα που αναζητάτε.

Με αυτή τη νέα λειτουργία έχετε 3 επιλογές:

- 1) **Φτιάξτε το πρόγραμμά σας 100% σε BASIC** (δηλ. μην χρησιμοποιείτε τη λειτουργικότητα). Αυτή η επιλογή απλοποιεί πολύ την εργασία προγραμματισμού, αλλά έχετε λιγότερη ταχύτητα.
- 2) **Κάντε το πρόγραμμά σας 100% σε C**. Αυτή η επιλογή είναι πολύπλοκη επειδή ο προγραμματισμός, η μεταγλώττιση, η αναζήτηση και η διόρθωση λαθών είναι πολύ πιο αργή εργασία από τον προγραμματισμό σε BASIC.
- 3) **Φτιάξτε το πρόγραμμά σας σε BASIC και στο τέλος μεταφράστε μόνο τον κύκλο του παιχνιδιού σε C**. Αυτή η επιλογή είναι εξίσου εύκολη με την πρώτη, εκτός από την τελευταία φάση της μετάφρασης σε C.

Θα σας εξηγήσω την τρίτη επιλογή, η οποία είναι πολύ ενδιαφέρουσα, διότι σας επιτρέπει να προγραμματίσετε σε BASIC με την ευκολία που έχει (εύκολο προγραμματισμό, εντοπισμό και διόρθωση λαθών κ.λπ.). Αν θέλετε να προγραμματίσετε το παιχνίδι σας εξ ολοκλήρου σε C, αυτή η εξήγηση σας εξυπηρετεί ακριβώς το ίδιο, οπότε μη φοβάστε και συνεχίστε να διαβάζετε. Ένα εμπορικό παράδειγμα της επιλογής 3 είναι το διάσημο και μυθικό παιχνίδι "galactic plague", μια παραγωγή της Indescomp που δημιουργήθηκε από τον εξαιρετικό προγραμματιστή Paco Suárez το 1984. Οφείλουμε πολλές ώρες ψυχαγωγίας στον σπουδαίο Paco Suárez.



Εικ. 112 Η "γαλαξιακή πανούκλα".

Για να μπορέσουμε να προγραμματίσουμε σε C χρειαζόμαστε κάποια εργαλεία, αλλά μην ανησυχείτε δεν χρειάζεται να μάθετε πώς να τα χρησιμοποιείτε γιατί το 8BP το κάνει για εσάς, χάρη σε ένα .bat που αναλαμβάνει τα πάντα, όπως θα δείτε σύντομα. Τα εργαλεία είναι τα εξής:

- **compila.bat**: είναι το αρχείο .bat που καλεί τα διάφορα εργαλεία του cascade και από ένα αρχείο .c λαμβάνετε ένα αρχείο .dsk με ένα δυαδικό αρχείο μέσα.

- Θα βρείτε αυτό το εργαλείο στον υποκατάλογο "C" του 8BP.
SDCC (Small Device C compiler): πρέπει να τον κατεβάσετε και να τον εγκαταστήσετε. Θα τον βρείτε στη διεύθυνση <http://sdcc.sourceforge.net/>. Πρόκειται ενδεχομένως για τον καλύτερο μεταγλωττιστή C για Z80 (αν και ο SDCC υποστηρίζει και άλλους μικροεπεξεργαστές). Υπάρχει ένας άλλος

Επέλεξα το Z88dk, αλλά προτίμησα να επιλέξω το SDCC, επειδή ορισμένες δοκιμές αποκαλύπτουν ότι το πρόγραμμα που προκύπτει από τη μεταγλώττιση με το SDCC είναι ταχύτερο από το αντίστοιχο του z88dk.

- **Hex2bin.exe:** θα το χρησιμοποιήσουμε (από το .bat) για να μετατρέψουμε το αρχείο που δημιουργήσαμε σε δεκαεξαδικό με το SDCC σε δυαδικό. Δεν χρειάζεται να το κατεβάσετε, είναι μικρό και θα το βρείτε στον υποκατάλογο "C" του 8BP.
- **ManageDsk.exe:** θα το χρησιμοποιήσουμε (από το .bat) για να τοποθετήσουμε το μεταγλωττισμένο δυαδικό μας αρχείο σε ένα αρχείο .dsk. Δεν χρειάζεται να το κατεβάσετε. Είναι μικρό και θα το βρείτε στον υποκατάλογο "C" του 8BP.

Ας δούμε τα απαραίτητα βήματα με ένα παράδειγμα και στη συνέχεια θα σας περιγράψω λεπτομερώς πώς να καλείτε κάθε μία από τις συναρτήσεις της 8BP από τη C, καθώς και τις νέες εντολές "minibasic" που σας παρέχει η 8BPV40 για να προγραμματίζετε στη C σαν να ήσασταν σε BASIC.

20.1 Πρώτο βήμα: προγραμματίστε το παιχνίδι BASIC

Θα χρησιμοποιήσουμε το πρόγραμμα παράδειγμα που συνοδεύει τη βιβλιοθήκη 8BP. Πρόκειται για ένα πολύ απλό παιχνίδι στο οποίο παίζετε έναν στρατιώτη που πρέπει να αποφύγει μπάλες που πέφτουν από τον ουρανό προς διάφορες κατευθύνσεις. Κάθε φορά που σας χτυπάει μια μπάλα, χάνετε πόντους, οι οποίοι αυξάνονται με την πάροδο του χρόνου.



Σχ. 113 Το σύνολο του παραδείγματος

Ο κατάλογος του παιχνιδιού έχει ως εξής, στον οποίο έχω επισημάνει με κόκκινο χρώμα το μέρος που αντιστοιχεί στον "κύκλο παιχνιδιού".

```

10 MNHMH 19999
11 LOAD "loop.bin",20000: REM φορτώστε τον βρόχο του παιχνιδιού που έχει μεταγλωττιστεί
20 MODE 0: DEFINT A-Z: CALL &6B78:' install RSX
30 ΣΤΟ ΔΙΑΛΕΙΜΜΑ GOSUB 320
40 CALL &BC02:'επαναφορά της προεπιλεγμένης παλέτας για παν ενδεχόμενο'.
50 INK 0,0: "μαύρο φόντο".
60 FOR j=0 TO 31:|SETUPSP,j,0,0:0:NEXT:|3D,0:'reset sprites
70 |SETLIMITS,0,80,0,124: ' ορίστε τα όρια της οθόνης του παιχνιδιού
80 PLOT 0,74*2:DRAW 640,74*2
90 x=40:y=100:' συντεταγμένες του χαρακτήρα
100 PRINT "SCORE:      FPS:"
110 |SETUPSP,31,0,1+32:' κατάσταση χαρακτήρων
120 |SETUPSP,31,7,1'Ακολουθία κινούμενων σχεδίων που

```

εκχωρείται κατά την εκκίνηση

130 |LOCATESP,31,y,x:'τοποθετήστε το sprite (χωρίς να το εκτυπώσετε ακόμα)

140 |MUSIC,0,0,0,0,0,0.5: points=0

150 cor=32:cod=32:|COLSPALL,@cor,@cod:' Ορισμός εντολής σύγκρουσης LOCATE 1,20:INPUT "basic(1) ή C(2)", a: IF a=1 THEN 160 ELSE CALL &56b0 GOTO 320

160 |PRINTSPALL,0,0,0,0,0,0: 'διαμόρφωση εντολής εκτύπωσης

161 POKE &B8B4,0: POKE &B8B5,0: POKE &B8B6,0: POKE &B8B7,0:'reset timer cpc6128. είναι απαραίτητη, καθώς το TIME μπορεί να επιστρέψει έναν πολύ μεγάλο αριθμό και να δώσ υπερχείλιση με DEDFINT τε t1=XPONOΣ

162

170 '--- κύκλος παιχνιδιού. Αυτό είναι το μέρος που έχει μεταφραστεί σε C
c=c+1

190 ' διαβάζει το πληκτρολόγιο και τοποθετεί τον χαρακτήρα

191 AN INKEY(27)=0 TOTE AN dir>>0 TOTE |SETUPSP,31,7,1:dir=0 ELSE

|ANIMA,31:x=x+1:GOTO 195

192 EAN INKEY(34)=0 TOTE EAN dir>>1 TOTE |SETUPSP,31,7,2:dir=1 ELSE

|ANIMA,31:x=x-1

195 |LOCATESP,31,y,x

200 |AUTOALL:|PRINTSPALL

210 |COLSPALL

220 IF cod<32 THEN BORDER 7:dots=dots-1:LOCATE 7,1:PRINT σημεία:GOTO

221 REM για τον υπολογισμό των FPS λαμβάνουμε υπόψη ότι ο XPONOΣ μου δίνει σε μονάδες 1/300 δευτερόλεπτα και θα μετράω κάθε 20 κύκλους. Επομένως, fps= 20 κύκλοι x 300 / dt, όπου dt= t2-t1

230 IF c MOD 20=0 THEN dots=dots+10 :LOCATE 7,1:PRINT dots:
t2=t1:t1=TIME:fps=6000/(t1-t2):LOCATE 17,1:PRINT fps

240 EAN c MOD 5=0 THEN |SETUPSP,i,9,9,19:|SETUPSP,i,5,4,RND*3-1:|SETUPSP,i,0,11:|LOCATESP,i,10,RND*80: i=i+1:IF i=30 THEN i=0 AN c<500 TOTE GOTO 180

251 '--- κύκλος τέλους παιχνιδιού ---

252 |POKE,42038,points

310 τέλος του παιχνιδιού

320 |MUSIC:INK 0,0:PEN 1:BORDER 0:|PEEK,42038,@dots

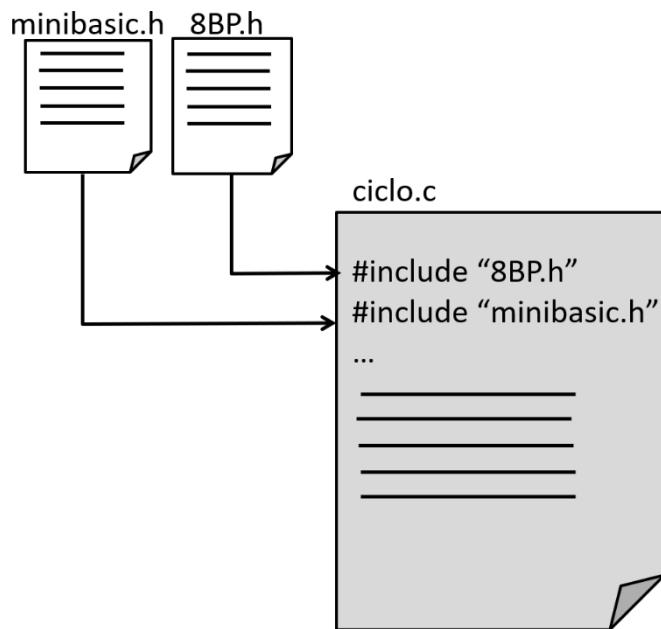
330 LOCATE 3,10:PRINT "FINAL SCORE:";dots

20.2 Δεύτερο βήμα: μεταφράστε τον κύκλο παιχνιδιού BASIC σε C

Για να μεταφράσουμε τον κύκλο του παιχνιδιού σε C, πρέπει να γράψουμε ένα πρόγραμμα C, το οποίο θα ονομάσουμε cycle.c.

Μεταβείτε στον υποφάκελο "C" της 8BP. Εκεί θα βρείτε όλα όσα χρειάζεστε και το ίδιο παράδειγμα, με το αρχείο ciclo.c.

Το πρώτο πράγμα που πρέπει να ξέρετε όταν προγραμματίζετε τον κύκλο του παιχνιδιού σε C, είναι ότι πρέπει να συμπεριλάβετε δύο μικρές βιβλιοθήκες: το περιτύλιγμα 8BP (8BP.h) και τη minibasic (minibasic.h). Η 8BP.h βρίσκεται στον υποκατάλογο "8BP_wrapper" και η minibasic.h στον υποκατάλογο "mini_basic".



Σχ. 114 αρχεία προς μεταγλώττιση

Αυτή είναι η λίστα της C, η οποία, όπως μπορείτε να δείτε, έχει άμεση αντιστοιχία με το κομμάτι της λίστας της BASIC **που αντιστοιχεί στον κύκλο του παιχνιδιού**, πρακτικά μια κυριολεκτική μετάφραση. Θα δείτε κάποιες ετικέτες όπως "label_195" που λειτουργούν ως αριθμοί γραμμών για να μπορείτε να μεταπηδήσετε με το GOTO. Πρόκειται πρακτικά για τη λίστα BASIC μεταφρασμένη εντολή προς εντολή, χωρίς να χρειάζεται να ξανασκεφτείτε πώς να την προγραμματίσετε. Σε αυτή την απλή περίπτωση υπάρχει μόνο μία συνάρτηση (η συνάρτηση main) και επιστρέφει όταν τελειώνει ο χρόνος.

Για να κάνετε αυτό το βήμα έχετε το "minibasic" για να σας βοηθήσει στη μετάφραση από τη BASIC στη C, αλλά αν είστε ειδικός στη C και γνωρίζετε το firmware AMSTRAD ή έχετε κάποια άλλη βοηθητική βιβλιοθήκη μπορείτε να κάνετε τον κύκλο του παιχνιδιού απευθείας στη C, χωρίς να το έχετε προγραμματίσει και επικυρώσει προηγουμένως στη BASIC. Ο τρόπος που σας προτείνω είναι εύκολος και ισχυρός, το πρόγραμμά σας θα τρέχει σαν λαγωνικό, αλλά εδώ είστε ελεύθεροι να το κάνετε όπως θέλετε.

ΣΗΜΑΝΤΙΚΟ: να θυμάστε όταν προγραμματίζετε σε C ότι η σημειογραφία για δεκαδικούς, δεκαεξαδικούς και δυαδικούς αριθμούς είναι:

mivariable = 165 ; //δεκαδική γραφή
mivariable = 0xA5 ; //εξαδική γραφή
mivariable = 0b10100101; //δυαδική γραφή

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <stdio.h>

#include "8BP.h"
#include "minibasic.h"
#include "8BP.h"
#include "minibasic.h"

//δήλωσε τις μεταβλητές όλες παγκόσμιες για να μπορείς να
//προσπέλαση από οποιαδήποτε συνάρτηση, όπως στη BASIC
// αν και δεν αρχικοποιούνται εδώ
//-----
int c,
```

```

char dir-
int x-
int y-
int cod-
int cor-
int i,
int points; int
t1,
int t2;
int fps,
*****
MAIN
*****
int main()
{
    //αρχικοποίηση των
    μεταβλητών c=0,
    dir=0;
    x=40;
    y=100,
    cod=32;
    cor=32;
    i=0,
    points=0;
    fps=0,
    t1=_basic_time(),

    //configure commands
    _8BP_printspall_4(0,0,0,0);
    _8BP_colspall_2(&cor,&cod),

    //κύκλος παιχνιδιού
    //-----
    label_CICLE:
    c=c+1,

    if (_basic_inkey(27)==0) {
        if (dir !=0) {
            _8BP_setupsp_3(31,7,1);
            dir=0,
        }
        else {
            _8BP_anima_1(31);
            x=x+1,
            goto label_195,
        }
    }
    if (_basic_inkey(34)==0) {
        if (dir !=1) {

```

```

    _8BP_setupsp_3(31,7,2);
    dir=1,
}
else {
    _8BP_anima_1(31);
    x=x-1,
}
}

label_195:
//-----
_8BP_locatesp_3(31,y,x);
_8BP_autoall(),
_8BP_printspall(),
_8BP_colspall(),

if (cod<32) {
    _basic_border(7),
    _basic_sound(1,100,14,0,0,1,0);
    puntos=puntos-1;
    _8BP_setupsp_3(cod,0,9);
    _basic_locate(7,1),
    _basic_print(_basic_str(dots)); goto
    _label_250,
}

else _basic_border(0);
if (c %20 ==0) {
    σημεία=σημεία+10,
    _basic_locate(7,1),
    _basic_print(_basic_str(points));
    t2=t1;t1= _basic_time();
    fps=6000/(t1-t2),
    _basic_locate(17,1);_basic_print(_basic_str(fps)),
}

if (c %5 ==0){
    _8BP_setupsp_3(i,9,19);
    _8BP_setupsp_4(i,5,4,_basic_rnd(3)-1);

    _8BP_setupsp_3(i,0,11);_8BP_locatesp_3(i,10,_basic_rnd(80));
    i=i+1;if (i==30) i=0,
}

_label_250:
if (c<500 goto label_CICLE,

_8BP_poke_2(42038,points);
return 0,

```

}

Όπως μπορείτε να δείτε, οι συναρτήσεις 8BP καλούνται ως εξής:

8BP<function>_<N>(parameters)

Ν είναι ο αριθμός των παραμέτρων της συνάρτησης, καθώς υπάρχουν εκδόσεις κάθε συνάρτησης με διαφορετικό αριθμό παραμέτρων (όπως συμβαίνει στις εκδόσεις RSX των εντολών).

Και οι βασικές συναρτήσεις που δεν είναι διαθέσιμες στη C αλλά που έχουμε δημιουργήσει στη minibasic καλούνται ως εξής:

basic<function>(parameters)

Όπως μπορείτε να δείτε, γνωρίζοντας τη μετάφραση κάθε εντολής είναι πολύ εύκολο να μεταφράσετε μια λίστα BASIC του κύκλου του παιχνιδιού σας σε μια λίστα cycle.c.

Ένα απαραίτητο πράγμα που πρέπει να κάνετε είναι να επικοινωνήσετε το LOCOMOTIVE BASIC με το C. Για παράδειγμα, υπάρχουν δεδομένα που μπορεί να θέλετε να περάσετε στο πρόγραμμα C, όπως ο αριθμός των ζωών που σας έχουν απομείνει ή οι πόντοι που έχετε συγκεντρώσει. Για αυτό μπορείτε να χρησιμοποιήσετε τις συναρτήσεις:

- Από τη LOCOMOTIVE BASIC έχετε τις PEEK, POKE, |PEEK και |POKE.
- Από τη C έχετε τις _basic.Peek(), _basic.Poke(), _8BP.Peek_20(),
_8BP.Poke_20()

Με αυτές τις συναρτήσεις μπορείτε να δεσμεύσετε μια διεύθυνση μνήμης για να αποθηκεύσετε τις ζωές, τους πόντους, τη φάση κ.λπ. και έτσι να επικαλεστείτε τον κύκλο του παιχνιδιού κάθε φορά που σκοτώνεστε με όλο το πλαίσιο του παιχνιδιού σε μερικές μεταβλητές που μπορούν να διαβαστούν και να τροποποιηθούν τόσο από τη BASIC όσο και από τη C. Στο παράδειγμα μπορείτε να δείτε πώς γίνεται αυτό με τη μεταβλητή "πόντοι".

ΣΗΜΑΝΤΙΚΟ: ακόμη και αν ολόκληρο το πρόγραμμά σας είναι σε C, πρέπει να αρχικοποιήσετε την 8BP με μια κλήση CALL &6b78, διότι εκτός από την εγκατάσταση των εντολών RSX, η κλήση αυτή αρχικοποιεί επίσης τους πίνακες της εσωτερικής βιβλιοθήκης.

20.2.1 GOSUB και RETURN στη C

Οι GOSUBs πρέπει να μεταφράζονται σε συναρτήσεις C, επειδή μπορείτε να φτάσετε σε μια ρουτίνα GOSUB από οποιδήποτε στο πρόγραμμα και πρέπει να μπορείτε να επιστρέψετε. Στη BASIC επιστρέφετε με RETURN στο σημείο από το οποίο καλέσατε το GOSUB. Στη C το φυσικό είναι να τη μεταφράσετε σε συνάρτηση, για να μπορείτε να επιστρέψετε στο σημείο του προγράμματος από το οποίο την καλέσατε

Ας δούμε ένα παράδειγμα:

BASIC

C

<pre> 10 a=5 20 GOSUB 100 30 PRINT "PEPE" 40 τξλος POYTINA 100 REM 110 PRINT STR\$(a) 120 ΕΠΙΣΤΡΟΦΗ </pre>	<pre> #include <stdlib.h> #include <string.h> #include <stdio.h> #include <stdio.h> #include "8BP.h" #include "minibasic.h" #include "8BP.h" #include "minibasic.h" </pre>
	<pre> void mifucion(int id); int a, int main() { a=5; mifunction(a), _basic_print(_"PEPE"), επιστροφή 0, } void mifucion(int a) { _basic_print(_basic_str(a)), _basic_print("\r"), } </pre>

20.2.2 Επικοινωνία BASIC προς C με μεταβλητές BASIC

Αν μόλις αρχίζετε να χρησιμοποιείτε τη C με την 8BP, μπορείτε να προχωρήσετε στο επόμενο βήμα. Αυτή είναι μια "προχωρημένη" ενότητα για να σας διδάξει πώς να επικοινωνείτε μεταξύ BASIC και C με μεταβλητές BASIC αντί για PEEK/POKE, αλλά δεν είναι απαραίτητη.

Για να επικοινωνήσετε τη BASIC με τη C, αντί για μια διεύθυνση μνήμης και PEEK/POKE, μπορείτε επίσης να χρησιμοποιήσετε μια μεταβλητή που υπάρχει στη BASIC. Είναι λίγο πιο περίπλοκο, αλλά μπορεί να γίνει. Ας δούμε πώς να το κάνουμε με μια απλή μεταβλητή, και στη συνέχεια θα δούμε πώς να το κάνουμε με μεταβλητές συστοιχίας.

Το πρώτο πράγμα που πρέπει να ξέρετε είναι πώς να βρείτε πού αποθηκεύει η BASIC μια μεταβλητή. Για το σκοπό αυτό έχουμε τον τελεστή "@". Ας δούμε ένα απλό παράδειγμα:

10 DEFINT A-Z: "σημαντικό για τον τύπο δεδομένων να είναι int".

20 a=5

30 print @a: 'εκτυπώνει τη διεύθυνση μνήμης όπου είναι αποθηκευμένο το a

40 poke @a,7: 'αυτό είναι το ίδιο με το να κάνεις a=7

50 PRINT a : ' αυτό εκτυπώνει a 7

Υπάρχει ένα μικρό "λάθος" στο πρόγραμμα. Είναι η εντολή POKE @a,7 επειδή αποθηκεύει μόνο ένα byte και η μεταβλητή "a" έχει 2 bytes επειδή είναι ακέραιος αριθμός. Αυτή η περίπτωση λειτουργεί επειδή το πιο σημαντικό byte της "a" είναι μηδέν, αλλά αν η "a" είχε τιμή μεγαλύτερη από 255 τότε το πιο σημαντικό byte της δεν θα ήταν μηδέν και η POKE θα άλλαζε μόνο το λιγότερο σημαντικό byte. Ένα 8BP POKE θα λειτουργούσε πάντα επειδή είναι 16 bit:

| **POKE,@a,7:** 'βάλτε ένα 7 στη μεταβλητή "a".

```
a=1000
Ready
print a
1000
Ready
print @a
432
Ready
poke @a,7
Ready
print a
775
Ready
```

Γνωρίζοντας αυτό, το μόνο που χρειάζεται να δώσουμε στη C είναι η διεύθυνση στην οποία είναι αποθηκευμένη η μεταβλητή BASIC. Για το σκοπό αυτό έχουμε την εντολή |POKE του 8BP που λειτουργεί με 16 bits (έχετε υπόψη σας ότι μια διεύθυνση μνήμης καταλαμβάνει 16 bits). Θα περάσουμε τη διεύθυνση της μεταβλητής "a" στη διεύθυνση 40000, αν και θα μπορούσαμε να χρησιμοποιήσουμε οποιαδήποτε άλλη ελεύθερη διεύθυνση.

|POKE, 40000, @a:'αφήνουμε το @a στη διεύθυνση 40000

Μια εναλλακτική μέθοδος στη BASIC είναι η χρήση δύο POKEs:

dir=@a:

POKE 40001, INT (dir/256) POKΕ

40000, INT (dir MOD 256)

Αυτό που γράψαμε στη διεύθυνση 40000 είναι η διεύθυνση μνήμης όπου είναι αποθηκευμένη η μεταβλητή a.

ΣΗΜΑΝΤΙΚΟ: Το BASIC μπορεί να μετατοπίσει μια μεταβλητή όταν δημιουργούνται νέες μεταβλητές. Αυτό σημαίνει ότι αν περάσετε σε μια ρουτίνα C μια διεύθυνση μνήμης μέσω ενός |POKE και στη συνέχεια δημιουργήσετε νέες μεταβλητές BASIC, η διεύθυνση μνήμης της μεταβλητής που περάσατε μπορεί να έχει αλλάξει. **Είναι εγγυημένο ότι δεν θα αλλάξει μόνο αν δεν δημιουργηθούν νέες μεταβλητές.** Το παρακάτω παράδειγμα το δείχνει πολύ καλά αυτό, υπάρχουν 2 αλλαγές θέσης

<pre>10 DIM a(100): i=0 20 PRINT @a(0) 30 b=2:'νέα μεταβλητή μετατοπίζει a() 40 PRINT @a(0) 50 c=2:'νέα μεταβλητή μετατοπίζει a() 60 PRINT @a(0) 70 goto 20</pre>	
---	--

Η λύση σε αυτό το πρόβλημα είναι να δηλώνονται όλες οι μεταβλητές στην αρχή του προγράμματος.

<pre>10 dim a(100) 20 b=2 30 c=3 40 print @a(0) 70 goto 40</pre>	
--	--

Τώρα από τη C μπορούμε να έχουμε πρόσβαση στη μεταβλητή "a" με δύο τρόπους, αν και ο δεύτερος τρόπος (μέσω μιας "αντιστοιχισμένης" μεταβλητής C) είναι ο πιο ενδιαφέρων.

```
// παγκόσμιες  
μεταβλητές  
int data,  
  
int main()  
{  
//ας αποθηκεύσουμε στη dir_a τη διεύθυνση της μεταβλητής a  
_8BP.Peek_2(40000, &dir_a),  
_8BP.Peek_2(dir_a, &data); //αυτό μεταφέρει την τιμή της μεταβλητής  
BASIC "a" στη μεταβλητή C "data". Αυτός είναι ένας τρόπος για να  
διαβάσετε την τιμή της "a".  
_8BP.Poke_2(dir_a,7); //αυτό βάζει ένα 7 στη μεταβλητή "a" της BASIC.  
επιστροφή 0,  
}
```

Ας δούμε το ίδιο παράδειγμα με άλλο τρόπο, με μια μεταβλητή C που "αντιστοιχίζεται" στη βασική μεταβλητή. Για αυτό πρέπει να χρησιμοποιήσουμε τον συμβολισμό με "*".

```
// παγκόσμιες
μεταβλητές
int *a,
int main()
{
//ας αποθηκεύσουμε στη dir_a τη διεύθυνση της μεταβλητής a
_8BP_peek_2(40000, &dir),
a=dir; //για να αποφύγετε την προειδοποίηση μεταγλώττισης
χρησιμοποιήστε a=(int*)dir
*a=5; //αυτό βάζει ένα 5 στη μεταβλητή "a" της BASIC.
επιστροφή 0,
}
```

Είδαμε πώς λειτουργεί με απλές μεταβλητές. Τώρα θα δούμε πώς λειτουργούν οι πίνακες της BASIC για να τους προσπελάσουμε από τη C. Το πρώτο πράγμα που θα κάνουμε είναι να καταλάβουμε πώς η BASIC αποθηκεύει τα δεδομένα των πινάκων στη μνήμη.

```
1 DEFINT a-z
2 ΛΕΙΤΟΥΡΓΙΑ 2
10 a=5
20 PRINT "το dir του a είναι
@a;@a 25 PRINT "-----"
30 DIM b(5)
40 FOR i=0 TO 5
50 PRINT "το dir του b(";i;")
είναι ";@b(i)
60 ΕΠΟΜΕΝΟ
70 PRINT "-----"
80 DIM c(3,4)
90 FOR j=0 TO 4:FOR i=0 TO 3
100 PRINT "το dir του c(";i;",";j;")
είναι ";@c(i,j)
110 ΕΠΟΜΕΝΟ:ΕΠΟΜΕΝΟ
120 |POKE,40000,@c(0,0)
```

Με τη γραμμή 120 έχουμε αποθηκεύσει στη διεύθυνση 40000, τη διεύθυνση μνήμης όπου αποθηκεύονται τα πρώτα δεδομένα του δισδιάστατου πίνακα.

Θα μπορούσαμε να αποθηκεύσουμε αυτό του μονοδιάστατου πίνακα με |POKE,40000, @b(0)

```
la dir de a es @a= 681
-----
la dir de b( 0 ) es 698
la dir de b( 1 ) es 700
la dir de b( 2 ) es 702
la dir de b( 3 ) es 704
la dir de b( 4 ) es 706
la dir de b( 5 ) es 708
-----
la dir de c( 0 , 0 ) es 727
la dir de c( 1 , 0 ) es 729
la dir de c( 2 , 0 ) es 731
la dir de c( 3 , 0 ) es 733
la dir de c( 0 , 1 ) es 735
la dir de c( 1 , 1 ) es 737
la dir de c( 2 , 1 ) es 739
la dir de c( 3 , 1 ) es 741
la dir de c( 0 , 2 ) es 743
la dir de c( 1 , 2 ) es 745
la dir de c( 2 , 2 ) es 747
la dir de c( 3 , 2 ) es 749
la dir de c( 0 , 3 ) es 751
la dir de c( 1 , 3 ) es 753
la dir de c( 2 , 3 ) es 755
la dir de c( 3 , 3 ) es 757
la dir de c( 0 , 4 ) es 759
la dir de c( 1 , 4 ) es 761
la dir de c( 2 , 4 ) es 763
la dir de c( 3 , 4 ) es 765
Ready
```

Το παραπάνω πρόγραμμα μας διδάσκει πώς η BASIC αποθηκεύει μεταβλητές.

- Τα δεδομένα μονοδιάστατου πίνακα αποθηκεύονται όλα σε μια σειρά. Κάθε δεδομένο καταλαμβάνει 2 bytes επειδή εργαζόμαστε με ακέραιους αριθμούς.

Για παράδειγμα, στη BASIC πληκτρολογείτε:

```
DIM b(20)
|POKE,40000,@b
CALL <routine C>:'διεύθυνση όπου έχει συναρμολογηθεί η main()
```

ΕΚΤΥΠΩΣΗ b(8)

και από το C γράφετε:

```
int dir,  
int *b; //μεταβλητή με την οποία θα έχουμε πρόσβαση στον  
πίνακα BASIC int main(){  
    _8BP.Peek_2(40000, &dir),  
    b= dir; //b είναι ένας δείκτης και *b[] είναι μεταβλητές  
    *b[8]=5; //προσθέτε ένα 5 στη μεταβλητή  
    BASIC b(8) return 0,  
}
```

- Τα δεδομένα των δισδιάστατων πινάκων αποθηκεύονται διαδοχικά και οι μεταβολές της πρώτης διάστασης παράγουν διαδοχικά δεδομένα. Στο παράδειγμα με το **DIM(3,4)** τα δεδομένα (2,3) ακολουθούν το (1,3) αλλά τα δεδομένα (2,3) είναι $4 \times 2 = 8$ bytes πιο μακριά από τα δεδομένα (2,2), επειδή η πρώτη διάσταση του πίνακα είναι 4. **Ένα DIM (3,4) είναι ένας πίνακας με διαστάσεις (4, 5) επειδή μετράει και το μηδέν.** Μπορείτε να το ελέγξετε αυτό εκτυπώνοντας τα @c(3,2) και @c(2,2), θα δείτε ότι υπάρχει διαφορά 8. Η πρόσβαση στα δεδομένα από τη C μπορεί να γίνει με έναν απλό δείκτη, λαμβάνοντας υπόψη την πρώτη διάσταση κατά την πρόσβαση. Στο παρακάτω παράδειγμα έχω δημιουργήσει έναν πίνακα c(12,16) και για να προσπελάσω από τη C στη θέση (2,7) χρησιμοποιώ $2+7*13$, αφού η πρώτη διάσταση είναι $12+1 = 13$

Στη BASIC γράφετε:

```
DIM c(12,16)  
|POKE, 40000, @c  
CALL <routine C>: ' διεύθυνση ρουτίνας συναρμολόγησης  
ΕΚΤΥΠΩΣΗ c(2,7)
```

και από το γ γράφετε:

```
int dir,  
int *b; //μεταβλητή με την οποία θα έχουμε πρόσβαση στον  
πίνακα BASIC int main(){  
    _8BP.Peek_2(40000, &dir); //διαβάζετε το dir 40000 και  
    γράφετε στο dir c= dir; //c είναι ένας δείκτης και *c[] είναι  
    μεταβλητές c[2+7*13]=5; //προσθέτε ένα 5 στη μεταβλητή  
    BASIC c(2,7) return 0,  
}
```

Αν είστε προγραμματιστής της C, θα γνωρίζετε ότι στη C υπάρχουν διπλοί δείκτες που εκφράζονται με διπλό αστερίσκο (π.χ. `**c`). Εκ των προτέρων φαίνονται κατάλληλοι επειδή μπορείτε να αναφέρεστε σε δεδομένα με `c[x][y]`. Ωστόσο, χρησιμεύουν μόνο για να αποθηκεύετε δυναμικά τη δεσμευμένη μνήμη με τις εντολές "malloc" και "calloc" και απαιτούν μνήμη τόσο για τα δεδομένα όσο και για τους δείκτες σε κάθε σειρά. Αυτό σημαίνει ότι θα μπορούσατε να τα χρησιμοποιήσετε, αλλά θα έπρεπε να δεσμεύσετε μνήμη για τους δείκτες. Δεν τους συνιστώ.

20.2.3 Συμβολοσειρές κειμένου BASIC και C

Θα πρέπει να γνωρίζετε ότι ένα αλφαριθμητικό στη C είναι ένας απλός δείκτης σε char, ενώ μια μεταβλητή αλφαριθμητικού στη BASIC (για παράδειγμα `myvar$="hello"`) αποτελείται από έναν περιγραφέα 3 byte που περιέχει τη διεύθυνση μνήμης όπου βρίσκεται το αλφαριθμητικό και το μήκος του αλφαριθμητικού. Δηλαδή, **ένα**

αλφαριθμητικό της C και ένα αλφαριθμητικό της BASIC είναι διαφορετικά πράγματα.

Έτσι, αν θέλετε να εκτυπώσετε μια μεταβλητή που περιέχει μια συμβολοσειρά κειμένου, δεν μπορείτε να τη μεταβιβάσετε από τη BASIC στη C επειδή δεν είναι το ίδιο πράγμα. Μπορείτε όμως να εκτυπώσετε συμβολοσειρές κειμένου χωρίς προβλήματα αν αντί να τις περάσετε από τη BASIC, τις ορίσετε στη C, για παράδειγμα:

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <stdio.h>
#include "8BP.h"
#include "minibasic.h"
#include "8BP.h"
#include "minibasic.h"

//----globalvariables
char* cad; //Θα μπορούσε να αρχικοποιηθεί εδώ

//----λειτουργίες-----
int main(){
cad="έχετε αποτύχει στην αποστολή σας"; //το αρχικοποιούμε με μια φράση
_basic_print(cad); //εκτύπωση σε στυλ BASIC
_8BP_printat4(0,0,60,cad); //εκτύπωση 8BP style printat
return 0,
}
```

20.3 Τρίτο βήμα: Μεταγλώττιση χρησιμοποιώντας το "compila.bat".

Είμαστε έτοιμοι να κάνουμε το μεγάλο βήμα: μεταγλώττιση για τη δημιουργία ενός δυαδικού αρχείου AMSTRAD. Για αυτό το βήμα έχετε ένα βοηθητικό .bat, που ονομάζεται "compila.bat". Πριν κάνετε αυτό το βήμα πρέπει να έχετε εγκαταστήσει τον μεταγλωττιστή SDCC στον υπολογιστή σας, διαφορετικά θα αποτύχει. Μπορείτε να τον κατεβάσετε από τη διεύθυνση <http://sdcc.sourceforge.net/>.

Σημείωση: Στα windows10, η εκτέλεση του SDCC με απογοήτευση επειδή είναι εγκατεστημένο στα "Αρχεία προγράμματος" και το κενό διάστημα δεν φαίνεται να του ταιριάζει. Αν αυτή είναι η περίπτωσή σας, απλά εγκαταστήστε το σε έναν κατάλογο του οποίου το όνομα δεν περιλαμβάνει κενά.

Αφού εγκατασταθεί το SDCC, ανοίξτε ένα παράθυρο εντολών (αυτό το κάνει το σενάριο ses.bat) και πληκτρολογήστε compila.bat. Η οθόνη θα αλλάξει χρώμα: πράσινο αν όλα πήγαν καλά και κόκκινο αν υπάρχουν προβλήματα. Το σενάριο "compila.bat" εκτελεί τα ακόλουθα βήματα:

1. Διαγράφει τα αρχεία εξόδου της ίδιας της δέσμης ενεργειών (ciclo.dsk και ciclo.bin μεταξύ άλλων).
2. Προκαλεί το SDCC για τη μεταγλώττιση του προγράμματός σας
3. Μεταφράστε την έξοδο SDCC σε δυαδική μορφή χρησιμοποιώντας το εργαλείο hex2bin
4. Εισάγετε το παραγόμενο δυαδικό αρχείο σε ένα δίσκο που ονομάζετε cycle.dsk, χρησιμοποιώντας το εργαλείο manageDsk.

Μετά την εκτέλεση του "compila.bat", θα εμφανιστεί μια κόκκινη οθόνη στις ακόλουθες περιπτώσεις:

- Συντακτικά σφάλματα στο πρόγραμμα C
- Το αρχείο ciclo.dsk που πρόκειται να δημιουργήσει το compila.bat ανοίγει από το winape. Σε αυτή την περίπτωση πρέπει να συνδέσετε έναν άλλο δίσκο στο winape, ώστε το compila.bat να μπορεί να το αναδημιουργήσει.
- Σφάλματα μεταγλώττισης, όπως μια βιβλιοθήκη που προσπαθήσατε να χρησιμοποιήσετε και δεν συμπεριλάβατε, κ.λπ.

Σε περίπτωση συντακτικών σφαλμάτων της C ή άλλων σφαλμάτων της C ή αν το αρχείο ciclo.dsk που θα δημιουργήσει το compila.bat ανοιχτεί από το winape, θα εμφανιστεί μια κόκκινη οθόνη σφάλματος όπως αυτή:

```
8888 BBBB BBBBB PPPPPP
88 88 BB BB PP PP
88 88 BB BB PP PP
8888 BBBB BBBBB PPPPP
88 88 BB BB PP
88 88 BB BB PP
8888 BBBB BBBBB PPPPP
8 bits de poder . Un tributo al AMSTRAD CPC
Jose Javier Garcia Aranda 2016-2020

*****
*      compilacion con SDCC      *
*****
borramos los ficheros de compilacion anterior
*****



main.c : compilamos y linkamos, generando un main.ihx
*****
sdcc -mz80 --verbose --code-loc 20000 --data-loc 0 --no-std-crt0 --
BP_wrapper -Imini_BASIC ciclo.c
sdcc: Calling preprocessor...
sdcc: sdcpp -nostdinc -Wall -std=c11 -I"8BP_wrapper" -I"mini_BASIC"
SDCC_CHAR_UNSIGNED -D__SDCC_INT_LONG_REENT -D__SDCC_FLOAT_REENT -D__
__SDCC_VERSION_MINOR=0 -D__SDCC_VERSION_PATCH=0 -D__SDCC_REVISION=1
=1 -D__STDC_NO_THREADS_=1 -D__STDC_NO_ATOMICS_=1 -D__STDC_NO_VLA_
DC_UTF_16_=1 -D__STDC_UTF_32_=1 -isystem "C:\proyectos\proyectos09"
-isystem "C:\proyectos\proyectos09\personal\8BP\SDCC\bin..\incl"
sdcc: Generating code...
ciclo.c:36: syntax error: token -> 'puntos' ; column 8
ciclo.c:37: error 1: Syntax error, declaration ignored at 'fps'
ciclo.c:38: error 1: Syntax error, declaration ignored at 't1'
ciclo.c:41: syntax error: token -> '0' ; column 21
.
"+-----+
 "| HAY ERRORES DE COMPILACION! |
"+-----+"

C:\proyectos\proyectos09\personal\8BP\V40\PROYECTO_V40_clean\C>
```

Εικ. 115 Το Compila.bat διαμαρτύρεται ότι έχετε σφάλματα C

Σε περίπτωση που όλα πήγαιναν καλά, αυτό θα ήταν το αποτέλεσμα

```

transformamos el .ihx en un .bin
*****
hex2bin -output\ciclo.ihx
hex2bin v1.0.1, Copyright (C) 1999 Jacques Pelletier
Lowest address = 00004E20
Highest address = 00005A41

metemos el .bin en un disco de amstrad cpc
*****
managedsk -C -S"output\ciclo.dsk"
managedsk -L"output\ciclo.dsk" -I"output\ciclo.bin"/CICLO.BIN/BIN/20

*****
**          FIN DEL PROCESO
**  ASEGUrate DE QUE NO EXCEDES LA DIRECCION 24000
** es la (highest address) de la transformacion ihx en bin
**
** se ha generado ciclo.dsk y dentro esta ciclo.bin
**
** Pasos para cargarlo en el amstrad
** 1) carga o ensambla 8BP, con tus graficos, musica etc
** 2) carga tu juego BASIC
** 3) ejecuta LOAD "ciclo.bin", 20000
** para invocar a tu programa o rutina simplemente:
** call <direccion de main en fichero ciclo.map>
**
** Para mover ciclo.bin de ciclo.dsk a otro disco debes
** conocer su longitud:
** longitud=Highest address - Lowest address
** lo cargas desde ciclo.dsk
** LOAD "ciclo.bin", 20000
** Y salvas en el disco donde esta tu juego
** SAVE "ciclo.bin",b,20000,longitud
*****
C:\proyectos\proyectos09\_personal\8BP\V40\PROYECTO_V40_clean\C>
```

Εικ. 116 To Compila.bat παράγει μια πράσινη οθόνη: όλα πήγαν καλά.

Σε περίπτωση που εμφανιστεί μια πράσινη οθόνη, αυτό σημαίνει ότι όλα τα βήματα του compila.bat πήγαν καλά (borSDCC, θα έχετε πολύτιμες πληροφορίες στην οθόνη και στον υποκατάλογο εξόδου θα βρείτε τα αρχεία που χρειάζεστε:

- Cycle.dsk
- Cycle.map

20.4 Βήμα τέταρτο: έλεγχος των ορίων μνήμης

Το σενάριο compila.bat σας δείχνει στην πράσινη οθόνη του δύο πολύ πολύτιμες πληροφορίες: "χαμηλότερη διεύθυνση" και "υψηλότερη διεύθυνση". Αυτές είναι οι διευθύνσεις μνήμης όπου ξεκινά και τελειώνει το παραγόμενο δυαδικό αρχείο. Το σενάριο "compila.bat" χρησιμοποιεί τη διεύθυνση 20000 ως διεύθυνση μεταγλώττισης στην κλήση SDCC και αν το δυαδικό αρχείο που προκύπτει είναι πολύ μεγάλο, μπορεί να υπερβεί τη διεύθυνση 24000, καταστρέφοντας έτσι τη βιβλιοθήκη 8BP. Πρέπει να βεβαιωθείτε ότι η "υψηλότερη διεύθυνση" είναι μικρότερη από 24000. Εάν δεν είναι μικρότερη, πρέπει να τροποποιήσετε την κλήση SDCC στο σενάριο "compila.bat".

να μεταγλωττίσει αναθέτοντας μια διεύθυνση μικρότερη από 20000. Με αυτόν τον τρόπο το νέο σας δυαδικό και η βιβλιοθήκη 8BP δεν θα επικαλύπτονται. Πρέπει να τροποποιήσετε δύο γραμμές του σεναρίου compila.bat. Η πρώτη είναι αυτή που καλεί το SDCC. Είναι μια πολύ μεγάλη γραμμή. Πρέπει να αλλάξετε την παράμετρο 20000 στη νέα διεύθυνση

```
sdcc -mz80 --verbose --code-loc 20000 --data-loc 0 --no-std-crt0 --fomit-frame-pointer --opt-code-size -I8BP_wrapper -lmini_BASIC -o output/cycle.c
```

Η δεύτερη γραμμή που πρέπει να τροποποιήσετε είναι αυτή που καλεί την managedsk ώστε να είναι συνεπής με τη νέα διεύθυνση μνήμης. Αυτή είναι η γραμμή και όπως μπορείτε να δείτε, εμφανίζεται επίσης η διεύθυνση 20000.

```
managedsk -L "output "cycle.dsk" -l "output "cycle.bin"/CYCLE.BIN/BIN/2000000 -I "output\cycle.bin"/CICLO.BIN/BIN/20000 -S "output\ciclo.dsk" -l "output\ciclo.bin"/CICLO.BIN/BIN/20000
```

Προφανώς, αν το δυαδικό σας σύστημα ξεκινάει από το 20000, το πρόγραμμα BASIC θα πρέπει να ενσωματώσει ένα 19999 MEMORY. Αυτό θα αφαιρέσει 4KB από τον ελεύθερο χώρο σας, αλλά θα αποθηκεύσετε επίσης τις γραμμές BASIC που αντιστοιχούν στον κύκλο του παιχνιδιού, οπότε ένα πράγμα για ένα άλλο και είναι σαν να μην έχετε χάσει τίποτα.

Εάν η υψηλότερη διεύθυνση είναι μικρότερη από 24000, θα πρέπει να την προσαρμόσετε όσο το δυνατόν περισσότερο, δηλαδή όσο το δυνατόν πιο κοντά στις 24000, ώστε να μην σπαταλάτε μνήμη. Αυτό μπορεί να περιλαμβάνει (για παράδειγμα) τη χρήση της διεύθυνσης 21000 κατά την κλήση του SDCC. Κάντε το αυτό για να έχετε όσο το δυνατόν περισσότερη μνήμη για τη BASIC. Θα πρέπει να βάλετε μια MEMORY που να συμφωνεί με αυτή τη διεύθυνση στο πρόγραμμα BASIC σας. Για παράδειγμα, αν το δυαδικό σύστημα ξεκινάει από τη διεύθυνση 21000, θα πρέπει να ορίσετε μια MEMORY της τάξης του 20999.

Τελικά, ίσως χρειαστεί να τροποποιήσετε δύο γραμμές στο σενάριο "compila.bat" για να προσαρμόσετε τη διεύθυνση έναρξης της μεταγλώττισης, την οποία αρχικά είχα ορίσει σε 20000.

20.5 Βήμα 5: Εντοπίστε τη διεύθυνση της συνάρτησης που θα κληθεί από τη BASIC.

Μετά τη μεταγλώττιση με το "compila.bat", στον υποκατάλογο "output" θα έχετε λάβει ένα αρχείο που ονομάζεται ciclo.map. Σε αυτό το αρχείο πρέπει να βρείτε τη διεύθυνση μνήμης της συνάρτησης ή των συναρτήσεων που σκοπεύετε να καλέσετε από τη BASIC. Σε αυτό το παράδειγμα πρόκειται να καλέσουμε μόνο τη συνάρτηση main(), η οποία βρίσκεται στη διεύθυνση &56b0, όπως μπορείτε να δείτε σε αυτό το τμήμα του αρχείου ciclo.map

Θεσσαλονίκη	basic_paper
0000566B	basic_plot
00005682	basic_move
00005699	basic_draw
000056B0	main
00005920	abs
0000592C	strlen
0000593B	modchar
00005948	modint
00005954	moduchar

**Σχ. 117 η διεύθυνση κάθε συνάρτησης βρίσκεται στο
ciclo.map**

Αυτό σημαίνει ότι για να καλέσουμε τη συνάρτηση main() από τη BASIC κάνουμε απλά:

ΚΛΗΣΗ &56B0

Να είστε προσεκτικοί, διότι αν κάνετε οποιεσδήποτε αλλαγές στη φάση της μεταγλώττισης (τροποποίηση του κύκλου .c ή αλλαγές στις διευθύνσεις μνήμης του σεναρίου compila.bat) η διεύθυνση κάθε συνάρτησης μπορεί να αλλάξει κατά την εκ νέου μεταγλώττιση.

20.6 Βήμα 6: Συμπεριλάβετε το νέο δυαδικό αρχείο στο παιχνίδι σας .dsk

Έχετε ήδη δημιουργήσει ένα αρχείο cycle.dsk που περιέχει το αρχείο cycle.bin, το οποίο πρέπει να φορτώσετε για να καλέσετε την κύρια λειτουργία (ή/και όποιες άλλες λειτουργίες θέλετε). Για να έχετε τόσο το αρχείο όσο και το παιχνίδι σας στον ίδιο δίσκο, πρέπει να επιλέξετε το αρχείο cycle.dsk από το winape και απλά να φορτώσετε το αρχείο cycle.bin.

ΦΟΡΤΙΟ "CYCLE.BIN",20000

Στη συνέχεια, από το μενού winape επιλέγετε το δίσκο σας (όπου έχετε το παιχνίδι σας) και αποθηκεύετε αυτό το δυαδικό αρχείο

ΑΠΟΘΗΚΕΥΣΗ "CICLO.BIN", b, 20000, <μήκος>.

όπου μήκος =υψηλότερη διεύθυνση - χαμηλότερη διεύθυνση +1

Τώρα, στο αρχείο loader.bas πρέπει να φορτώσετε αυτό το νέο πρόσθετο δυαδικό αρχείο και να το καλέσετε από την καταχώριση BASIC με CALL <διεύθυνση>.

10 MNHMH 24999

15 LOAD "!pant.scr",&c000: 'μόνο αν το παιχνίδι σας έχει οθόνη φόρτωσης

20 LOAD "yourgame.bin"

25 LOAD "cycle.bin", 20000

50 RUN " !yourgame.bas"

Αυτό είναι όλο. Τώρα μπορείτε να προγραμματίσετε σε C με την 8BP!

20.7 Αναφορά λειτουργίας 8BP σε C

RSX	Πρωτότυπο C
3D, 0 3D, <flag>, #, offsety	void _8BP_3D_1(int flag), void _8BP_3D_3(int flag, int sp_fin, int offsety),
ANIMA, #	void _8BP_anima_1(int sp),
ANIMALL	void _8BP_animall(),
AUTO, #	void _8BP_auto_1(int sp),
AUTOALL, <flag routed>, <flag routed>, <flag routed>, <flag routed>, <flag routed>.	void _8BP_autoall(), void _8BP_autoall_1(int flag),
COLAY, threshold_ascii, @collision, # COLAY, @collision, # COLAY, # COLAY	void _8BP_colay_3(int threshold, int* collision, int sp), void _8BP_colay_2(int* collision, int sp), void _8BP_colay_1(int sp); void _8BP_colay(),
COLSP, #, @collided%, @collided%, @COLSP, #, @collided%, @collided%. COLSP, 32, ini, τέλος COLSP, 33, @collided% COLSP, # COLSP, 34, dy, dx	/* λειτουργία 32, ini,fin ή λειτουργία 34,dy,dx*/ void _8BP_colsp_3(int operation, int a, int b), /*λειτουργία 33 ή sp*/ void _8BP_colsp_2(int sp, int* collision); void _8BP_colsp_1(int sp),
COLSPALL,@who%,@who%,@with whom% COLSPALL, συγκρουστήρας COLSPALL	void _8BP_colspall_2(int* collider, int* collided); void _8BP_colspall_1(int collider_ini), void _8BP_colspall(),
LAYOUT, y, x, @string\$, @string\$, @string\$, @string\$, @string\$, @string\$, @string\$, @string\$.	void _8BP_layout_3(int y, int x, char* cad),
LOCATESP, #, y, x	void _8BP_locatesp_3(char sp, int y, int x),
MAP2SP, y, x MAP2SP, κατάσταση	void _8BP_map2sp_2(int y, int x), void _8BP_map2sp_1(unsigned char status),
MOVER, #, dy, dx	void _8BP_mover_3(int sp, int dy,int dx); void _8BP_mover_1(int sp),
MOVERALL, dy,dx	void _8BP_moverall_2(int dy, int dx); void _8BP_moverall(),
MUSIC, C, σημαία, τραγούδι, ταχύτητα MUSIC, σημαία, τραγούδι, ταχύτητα MUSIC	void _8BP_music_4(int flag_c, int flag_repetition,int song, int speed), void _8BP_music(),
PEEK, dir, @variable%	void _8BP_peek_2(int address, int* data),
POKE, dir, value	void _8BP_poke_2(int address, int data),
PRINTAT, σημαία, y, x, @string	void _8BP_printat_4(int flag,int y,int x,char* cad),
PRINTSP, #, y, x PRINTSP, # PRINTSP,32, bits	void _8BP_printsp_1(int sp) , void _8BP_printsp_2(int sp, int bits_background) ; void _8BP_printsp_3(int sp,int y,int x) ,
PRINTSPALL, ini, fin, anima, sync PRINTSPALL, ordermode PRINTSPALL	void _8BP_printspall_4(int ini, int fin, int flag_anima, int flag_sync), void _8BP_printspall_1(int order_type); void _8BP_printspall(),
RINK,tini,color1,color1,color2,...,color N RINK, άλμα	void _8BP_rink_N(int num_params,int* ink_list); void _8BP_rink_1(int step),
ROUTEESP, #, βήματα	void _8BP_routesp_2(int sp, int steps)- void _8BP_routesp_1(int sp),
ROUTEALL	void _8BP_routeall(),
SETLIMITS, xmin, xmax, ymin, ymax	Void _8BP_setlimits_4 (int xmin, int xmax, int ymin, int ymax)
SETUPSP, #, param_number, value SETUPSP, #, 5, Vy, Vx	void _8BP_setupsp_3(int sp, int param, int value)- void _8BP_setupsp_4(int sp, int param, int value1,int value2),

STARS, initstar, num, color, dy, dx	void _8BP_stars_5(int star_ini, int num_stars,int color, int dy, int dx), void _8BP_stars(),
UMAP,adr_ini, adr_end, yini, yfin, xini, xfin	void _8BP_umap_6(int map_ini, int map_fin, int y_ini, int y_fin, int x_ini, int x_fin),

Ακολουθούν ορισμένα παραδείγματα χρήσης, τα οποία συμπληρώνουν το παράδειγμα που χρησιμοποιήθηκε στη μετάφραση από BASIC σε C.

RSX	 3D, <flag>, #, offset 10 3D,1,10,200
C	void _8BP_3D_3(int flag, int sp_fin, int offset), _8BP_3D_3(1, 10, 200),

RSX	 COLSPALL,@who%,@who%,@withwhom% 10 collider%=0: collided%=0 20 COLSPALL, @collider%, @colliderd%, @colliderd%
C	void _8BP_colspall_2(int* collider, int* collided), Int cor=0; Inr cod=0, _8BP_colspall_2 (&cor, &cod),

RSX	 RINK,tini,color1,color2,...,colorN RINK, άλμα 10 RINK,1,2,2,2,2,3,3 20 RINK,1
C	void _8BP_rink_N(int num_params,int* ink_list); void _8BP_rink_1(int step), Int inks[5]={1,2,2,2,3,3}, _8BP_rink_N(5,μελάνια), _8BP_rink_1(1),

RSX	 LAYOUT, y, x, @string\$, @string\$, @string\$, @string\$, @string\$, @string\$, @string\$, @string\$. 10 cad\$="AA YYYYYYYY YYYYY YYYYY YYYYY YYYYY YYYYY YYYYY" 20 LAYOUT,10,1,@cad\$
C	void _8BP_layout_3(int y, int x, char* cad), _8BP_layout_3(10,1," YYYYYYYY YYYYY YYYYY YYYYY YYYYY YYYYY YYYYY", 'H: char* cad=" AA YYYYYYYY YYYYY YYYYY YYYYY YYYYY YYYYY YYYYY", _8BP_layout_3(10,1,cad)

RSX	 PRINTAT, σημαία, y, x, @string 10 cad\$=str\$(125) 20 PRINTAT,0,100,40,@cad\$ 20 PRINTAT,0,100,40,@cad\$ 20
C	void _8BP_printat_4(int flg,int y,int x,char* cad) char* cad=_basic_str(125), _8BP_printat_4(0,100,40, cad),

20.8 Παραπομπή σε συνάρτηση BASIC σε C ("minibasic")

Αυτό είναι το μικρό σύνολο εντολών που μοιάζουν με BASIC και το 8BP σας παρέχει μέσω της βιβλιοθήκης minibasic.h, για να μεταφράσετε εύκολα την καταχώριση BASIC σε C. Αν προσπαθείτε να μεταφράσετε μόνο τον κύκλο του παιχνιδιού, αντό το σύνολο εντολών θα είναι αρκετό.

BASIC	Πρωτότυπο C
BORDER	<code>void _basic_border(char color), //παράδειγμα _basic_border(7)</code>
ΚΑΛΕΣΤΕ	<code>void _basic_call(unsigned int address), // παράδειγμα _basic_call(0xbd19)</code>
DRAW	<code>void _basic_draw(int x, int y),</code>
INK	<code>void _basic_ink(char ink1,char ink2),</code>
INKEY	<code>char _basic_inkey(char key), //διαρκεί περίπου 0,3 ms. αργό αλλά απλό</code>
ΤΟΠΟΘΕΤΗΣ H	<code>void _basic_locate(unsigned int x, unsigned int y), // παράδειγμα: _basic_locate(2,25); _basic_print("TEST"),</code>
MOVE	<code>void _basic_move(int x, int y),</code>
XAPTI	<code>void _basic_paper(char ink),</code>
PEEK	<code>char _basic_peek(unsigned int address),</code>
GRAPHICS PEN	<code>void _basic_pen_graph(char ink),</code>
PEN	<code>void _basic_pen_txt(char ink),</code>
POKE	<code>void _basic_poke(unsigned int address, unsigned char δεδομένα),</code>
PLOT	<code>void _basic_plot(int x, int y),</code>
ΕΚΤΥΠΩΣΗ	<code>void _basic_print(char *cad), //παράδειγμα: _basic_print("Hello")</code>
RND	<code>unsigned int _basic_rnd(int max), //παράδειγμα: num=_basic_rnd(50)</code>
ΗΧΟΣ	<code>void _basic_sound(unsigned char nChannelStatus, int nTonePeriod, int nDuration, unsigned char nVolume, char nVolumeEnvelope, char nToneEnvelope, unsigned char nNoisePeriod),</code>
STR\$	<code>char* _basic_str(int num), //όμοια με το STR\$ //παράδειγμα: _basic_print(_basic_str(num))</code>
ΧΡΟΝΟΣ	<code>unsigned int _basic_time(), //επιστρέφει ένα unsigned int,(0..65535). Ως ακέραιος, όταν // φτάνουμε στο 32768 πηγαίνουμε στο -32768</code>

21 Οδηγός αναφοράς βιβλιοθήκης 8BP

21.1 Λειτουργίες βιβλιοθήκης

21.1.1 |3D

Αυτή η εντολή ενεργοποιεί την τρισδιάστατη προβολή στις εντολές PRINTSP και PRINTSPALL Για την προβολή έχουμε την εντολή |3D

Χρήση

Για να ενεργοποιήσετε την τρισδιάστατη προβολή:

|3D, 1, <Sprite_fin>, <offsety>, <offsety>, <offsety>, <offsety>, <offsety>, <offsety>.

Για να το απενεργοποιήσετε:

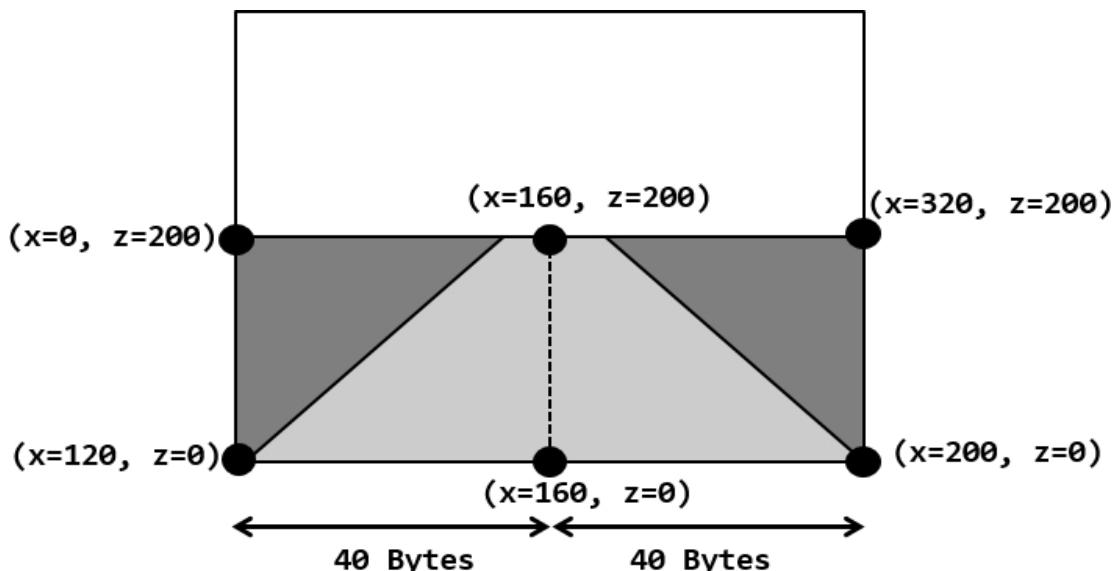
|3D, 0

Τα sprites που επηρεάζονται είναι από το Sprite 0 έως το <Sprite_fin>. Αυτή η εντολή ενεργοποιεί την τρισδιάστατη προβολή στην εντολή **|PRINTSP** και στην εντολή **|PRINTSPALL**. Αυτό σημαίνει ότι πριν από την εκτύπωση στην οθόνη, οι "προβαλλόμενες" συντεταγμένες θα υπολογιστούν και στη συνέχεια θα εκτυπωθούν στην οθόνη. Οι συντεταγμένες των sprites δεν επηρεάζονται, δηλαδή οι 2D συντεταγμένες στον πίνακα sprite θα παραμείνουν οι ίδιες.

Αυτή η εντολή **δεν επηρεάζει τους μηχανισμούς σύγκρουσης**, δηλαδή αν χρησιμοποιήσουμε COLSPALL και ανιχνεύσουμε μια σύγκρουση μεταξύ προβαλλόμενων sprites, η σύγκρουση συμβαίνει στο επίπεδο 2D.

Όσον αφορά την τελευταία παράμετρο <offsety> είναι να προβάλλουμε υψηλότερα ή χαμηλότερα, ώστε να μπορούμε να τοποθετήσουμε τους δείκτες του παιχνιδιού όπου θέλουμε. Κατά την προβολή της οθόνης, η οποία έχει ύψος 200 pixel, γίνεται 100 pixel ύψος, οπότε μπορούμε να επιλέξουμε πόσο ψηλά θα τοποθετήσουμε την προβολή. Αν ένα Sprite δεν επηρεάζεται από την προβολή επειδή είναι ψηλότερα από το <Sprite_fin>, τότε δεν επηρεάζεται ούτε από το <offsety>.

Το ακόλουθο σχήμα απεικονίζει τις συντεταγμένες του παγκόσμιου χάρτη που προβάλλονται σε ορισμένα αντιπροσωπευτικά σημεία της οθόνης όταν το **|MAP2SP** καλείται με (yo=0, xo=0).



Σχ. 118 προβαλλόμενες παγκόσμιες συντεταγμένες

Αν αντί για ($x_0=0, y_0=0$) χρησιμοποιήσουμε άλλη συντεταγμένη για το MAP2SP, οι συντεταγμένες του κόσμου 2D που αντιστοιχούν στα σημεία που αναφέρονται στην εικόνα θα μετατοπιστούν κατά (x, z) όπως υποδεικνύεται από τα χο και γο.

21.1.2 |ANIMA

Αυτή η εντολή αλλάζει το πλαίσιο κίνησης ενός sprite, λαμβάνοντας υπόψη την ακολουθία κίνησης που του έχει ανατεθεί.

Χρήση:

|ANIMA, <αριθμός αρίθμησης>, <αριθμός αρίθμησης>, <αριθμός αρίθμησης>, <αριθμός αρίθμησης>.

Παράδειγμα:

|ANIMA,3

Η εντολή ζητά την ακολουθία κίνησης του sprite και αν είναι μη μηδενική, τότε πηγαίνει στον πίνακα ακολουθιών κίνησης (η πρώτη έγκυρη ακολουθία είναι το 1 και η τελευταία το 31). Επιλέγει την εικόνα της οποίας η θέση είναι δίπλα στο τρέχον καρέ και ενημερώνει το πεδίο frame του πίνακα χαρακτηριστικών sprite.

Εάν το επόμενο πλαίσιο της ακολουθίας είναι μηδενικό, τότε γίνεται κυκλική διαδικασία, δηλαδή επιλέγεται το πρώτο πλαίσιο της ακολουθίας.

Εκτός από την αλλαγή του πεδίου πλαισίου, αλλάζει και το πεδίο εικόνας και εκχωρείται η διεύθυνση μνήμης όπου αποθηκεύεται το νέο πλαίσιο.

|ANIMA δεν εκτυπώνει το sprite, αλλά το αφήνει έτοιμο για εκτύπωση, έτσι ώστε να εκτυπωθεί το επόμενο πλαίσιο της ακολουθίας του.

|ANIMA δεν ελέγχει ότι η σημαία κίνησης είναι ενεργή στο byte κατάστασης του sprite. Στην πραγματικότητα, ο χαρακτήρας μας κανονικά θα θέλει να κινείται μόνο όταν κινείται και όχι πάντα όταν εκτυπώνεται.

Εάν η ακολουθία κινούμενων σχεδίων είναι μια "ακολουθία θανάτου" (περιλαμβάνει ένα "1" στο τελευταίο της καρέ), τότε μόλις φτάσει στο καρέ του οποίου η διεύθυνση μνήμης εικόνας είναι 1, το sprite θα γίνει ανενεργό.

Η βιβλιοθήκη 8BP σας επιτρέπει να φτιάχνετε "ακολουθίες θανάτου", δηλαδή ακολουθίες που, μετά την ολοκλήρωσή τους, το sprite μεταβαίνει σε ανενεργή κατάσταση. Αυτό υποδεικνύεται από ένα απλό "1" ως τιμή της διεύθυνσης μνήμης του τελευταίου πλαισίου. Αυτές οι ακολουθίες είναι πολύ χρήσιμες για τον ορισμό εκρήξεων εχθρών που κινούνται με |ANIMA ή |ANIMALL. Αφού τους χτυπήσετε με τη βολή σας, μπορείτε να τους συνδέσετε μια ακολουθία κινούμενων σχεδίων θανάτου και στους επόμενους κύκλους του παιχνιδιού θα περάσουν από τις διάφορες φάσεις κινούμενων σχεδίων της έκρηξης και όταν φτάσουν στην τελευταία θα περάσουν σε ανενεργή κατάσταση, χωρίς να εκτυπώνουν πλέον. Αυτή η ανενεργή κατάσταση γίνεται αυτόματα, οπότε αυτό που έχετε να κάνετε είναι απλά να ελέγξετε τη σύγκρουση της βολής σας με τους εχθρούς και αν συγκρουστεί με κάποιον από αυτούς να αλλάξετε την κατάσταση με |SETUPSP έτσι ώστε να μην μπορεί να συγκρουστεί πλέον και να του αναθέσετε την ακολουθία κινούμενων σχεδίων θανάτου, επίσης με |SETUPSP.

Αν χρησιμοποιήσετε μια ακολουθία θανάτου, μην ξεχάσετε να βεβαιωθείτε ότι το τελευταίο καρέ πριν βρείτε το "1" είναι εντελώς άδειο, ώστε να μην υπάρχει κανένα ίχνος της έκρηξης.

Παράδειγμα ακολουθίας θανάτου

```
dw EXPLOSION_1,EXPLOSION_2,Explosion_3,1,0,0,0,0,0,0,0,0
```

21.1.3 |ANIMALL

Αυτή η εντολή κινεί όλα τα sprites που έχουν τη σημαία κινούμενων σχεδίων ρυθμισμένη στο byte κατάστασης. Αυτή η εντολή δεν έχει παραμέτρους

Χρήση
|ANIMALL

ΣΗΜΑΝΤΙΚΟ: από την έκδοση v37 της βιβλιοθήκης, η εντολή αυτή είναι προσβάσιμη μόνο μέσω της εντολής CALL (βλέπε πίνακα αντιστοιχιών στο παράρτημα) και όχι μέσω της εντολής RSX. Η αφαίρεσή της από τη λίστα εξοικονόμησε μερικά bytes μνήμης και μπορεί ακόμα να χρησιμοποιηθεί είτε από μια παράμετρο στην PRINTSPALL είτε από μια κλήση CALL.

Συνιστάται αν πρόκειται να κάνετε animation σε πολλά sprites, καθώς είναι πολύ πιο γρήγορη από το να καλείτε την εντολή |ANIMA αρκετές φορές.

Καθώς κανονικά θα θέλετε να ενεργοποιείτε την εντολή |ANIMALL σε κάθε κύκλο παιχνιδιού, πριν από την εκτύπωση των sprites, υπάρχει ένας πιο αποτελεσματικός τρόπος για να την ενεργοποιήσετε, και αυτός είναι να ορίσετε την αντίστοιχη παράμετρο της εντολής |PRINTSPALL σε "1", δηλ.

|PRINTSPALL,1,0

Αυτή η συνάρτηση καλεί εσωτερικά το **|ANIMALL** πριν από την εκτύπωση των sprites, αποθηκεύοντας 1,17ms σε σύγκριση με το χρόνο που θα χρειαζόταν για να κληθούν ξεχωριστά τα **|ANIMALL** και **|PRINTSPALL**

21.1.4 |AUTO

Αυτή η εντολή μετακινεί ένα sprite (αλλάζει τις συντεταγμένες του) σύμφωνα με τα χαρακτηριστικά ταχύτητας Vy, Vx. Αυτά τα χαρακτηριστικά είναι ότι έχει το sprite στον πίνακα sprite.

Χρήση:

|AUTO, <αριθμός αρίθμησης>, <αριθμός αρίθμησης>, <αριθμός αρίθμησης>, <αριθμός αρίθμησης>.

Παράδειγμα:

|AUTO, 5

Αυτή η εντολή ενημερώνει τις συντεταγμένες στον πίνακα sprite, προσθέτοντας την ταχύτητα στην τρέχουσα συντεταγμένη.

Οι νέες συντεταγμένες είναι
new X = τρέχουσα συντεταγμένη X
+ Vx new Y = τρέχουσα συντεταγμένη Y + Vy

Δεν είναι απαραίτητο το sprite να έχει ενεργή τη σημαία αυτόματης κίνησης στο πεδίο κατάστασης.

21.1.5 |AUTOALL

Αυτή η εντολή μετακινεί όλα τα sprites που έχουν ενεργή τη σημαία αυτόματης κίνησης, σύμφωνα με τα χαρακτηριστικά ταχύτητας Vy, Vx.

Χρήση:

|AUTOALL, <σημαία δρομολόγησης>, <σημαία δρομολόγησης>.

Παράδειγμα

|AUTOALL,1 καλεί το |ROUTEALL πριν μετακινήσει τα sprites

|AUTOALL,0 δεν καλεί |ROUTEALL

|AUTOALL η τελευταία τιμή που χρησιμοποιήθηκε χρησιμοποιείται ως παράμετρος (έχει μνήμη)

Η σημαία δρομολόγησης είναι προαιρετική. Δεδομένου ότι η εντολή |ROUTEALL δεν τροποποιεί τις συντεταγμένες των sprites, αυτά πρέπει να μετακινηθούν με την εντολή |AUTOALL και να εκτυπωθούν (και να εμψυχωθούν) με την εντολή |PRINTSPALL. Γι' αυτό έχετε μια προαιρετική παράμετρο στην |AUTOALL, έτσι ώστε η |AUTOALL,1 να καλεί εσωτερικά την |ROUTEALL πριν μετακινήσετε το sprite, γλιτώνοντάς σας από μια κλήση της BASIC που θα διαρκεί πάντα ένα πολύτιμο χιλιοστό του δευτερολέπτου.

21.1.6 |COLAY

Ανιχνεύει τη σύγκρουση ενός sprite με το χάρτη οθόνης (τη διάταξη). Λαμβάνει υπόψη το μέγεθος του sprite για να διαπιστώσει αν συγκρούεται και θεωρεί ότι τα στοιχεία της διάταξης είναι όλα 8x8 pixel του mode 0 (δηλαδή 4 bytes x 8 γραμμές). Μπορεί να

κληθεί με 3,2,1 ή χωρίς παραμέτρους. Αν κληθεί χωρίς παραμέτρους, θα χρησιμοποιηθούν οι τιμές της τελευταίας κλήσης με παραμέτρους και είναι πολύ πιο γρήγορη.

Χρήση:

|COLAY, <κατώφλι ASCII>, @collision%, <αριθμός σύνθλιψης>,
<αριθμός σύνθλιψης>.
|COLAY, @collision%, <αριθμός στίξης>
|COLAY, <num_sprite>, <num_sprite>.
|COLAY

Η προαιρετική παράμετρος <ASCII threshold> χρειάζεται να χρησιμοποιηθεί μόνο σε μια πρώτη κλήση για να οριστεί το κατώφλι σύγκρουσης στην εντολή |COLAY|. Αυτό το κατώφλι αντιπροσωπεύει τον μεγαλύτερο κωδικό ASCII του στοιχείου διάταξης που θεωρείται ως "χωρίς σύγκρουση". Η προεπιλογή είναι 32 (το λευκό διάστημα). Για να ορίσετε το κατώφλι που θέλετε, ανατρέξτε στον πίνακα ASCII της εντολής |LAYOUT|.

Παράδειγμα:

|COLAY, 65, @col%,31 : rem to sprite είναι 31, το όριο είναι 65

Η μεταβλητή που χρησιμοποιείτε για τη σύγκρουση μπορεί να ονομάζεται όπως θέλετε. Εγώ έβαλα "col".

Αυτή η ρουτίνα τροποποιεί τη μεταβλητή σύγκρουσης (η οποία πρέπει να είναι ακέραιος αριθμός, εξ ου και το "%") θέτοντάς την σε 1 εάν υπάρχει σύγκρουση του υποδεικνύμενου sprite με τη διάταξη. Εάν δεν υπάρχει σύγκρουση, το αποτέλεσμα είναι 0.

10 xprevious=x

20 x=x+1

30 |LOCATESP,0,y,x: ' τοποθέτηση του sprite στη νέα θέση

40 |COLAY,@collision%,0: 'έλεγχος σύγκρουσης

Τώρα ελέγχουμε τη σύγκρουση και αν υπάρχει σύγκρουση το αφήνουμε στην προηγούμενη θέση του.

50 εάν collision%=1 τότε x=xprevious: LOCATESP,0,y,x

Μπορείτε επίσης να χρησιμοποιήσετε την εντολή |MOVER| για να τοποθετήσετε το sprite και να κάνετε τον έλεγχο.

10 |COLAY,65,@col,31: 'διαμόρφωση. Το κάνουμε μόνο μία φορά

20 |MOVER,31,1,1: ' το μετακινούμε προς τα δεξιά και προς τα κάτω

30 |COLAY: ' κλήση χωρίς παραμέτρους (πιο γρήγορα)

30 if col THEN MOVER,31,-1,-1 : το rem έχει συγκρουστεί και έτσι το αφήνω εκεί που ήταν.

21.1.7 |COLSP

Αυτή η εντολή επιτρέπει την ανίχνευση της σύγκρουσης ενός sprite με άλλα sprites που έχουν ενεργή τη σημαία σύγκρουσης.

Χρήση :

Για να ρυθμίσετε τις παραμέτρους:

- | COLSP, 32, <πρωτογενής αρχικός>, <πρωτογενής τελικός>.
- | COLSP, 33, @collision%
- | COLSP, 34, dy, dx

Για την ανίχνευση σύγκρουσης:
|COLSP,<αριθμός πρωτεύουσας>, @colsp%.

Παράδειγμα:

col%=0

|COLSP,0,@col%

Η συνάρτηση επιστρέφει στη μεταβλητή που περνάμε ως παράμετρο, τον αριθμό του sprite με το οποίο συγκρούεται, ή αν δεν υπάρχει σύγκρουση επιστρέφει ένα 32 επειδή το sprite 32 δεν υπάρχει (υπάρχουν μόνο από το 0 έως το 31).

ΣΗΜΑΝΤΙΚΟ: η μεταβλητή σύγκρουσης στην εντολή COLSP δεν είναι αυτή που χρησιμοποιείται στην εντολή COLSPALL. Πρόκειται για διαφορετικές μεταβλητές (εκτός αν δώσετε και στις δύο εντολές την ίδια μεταβλητή για να ενεργήσετε σε αυτήν).

Όπως και η εκτύπωση sprites με την PRINTSPALL, η συνάρτηση COLSP ελέγχει sprites που αρχίζουν από το 31 και τελειώνουν στο μηδέν. Εάν έχουν ενεργή σημαία σύγκρουσης sprite (bit 2 του byte κατάστασης), τότε ελέγχεται η σύγκρουση. Εάν δύο sprites συγκρουστούν ταυτόχρονα με το sprite μας, επιστρέφεται ο μεγαλύτερος αριθμός sprite καθώς είναι αυτός που ελέγχεται πρώτος.

Κλήσεις για τη διαμόρφωση της εντολής:

Υπάρχει ένας τρόπος να ρυθμίσετε το COLSP ώστε να κάνει λιγότερη δουλειά, ελέγχοντας λιγότερα sprites για να εξοικονομήσετε χρόνο εκτέλεσης. Η διαμόρφωση θα υποδεικνύεται από τη χρήση του sprite 32 (το οποίο δεν υπάρχει).

|COLSP, 32, <αναφέρετε το αρχικό προς έλεγχο>, <αναφέρετε το τελικό προς έλεγχο>.

Αν για παράδειγμα οι εχθροί του χαρακτήρα μας είναι τα sprites 25 έως 30 και τους ρυθμίσουμε ως colliders (όχι συγκρουόμενους) μπορούμε να καλέσουμε (μόνο μία φορά) την εντολή ως εξής:

|COLSP, 32, 25, 30

Αυτό σημαίνει ότι κάθε επόμενη κλήση της εντολής |COLSP θα πρέπει να ελέγχει μόνο τη σύγκρουση των sprites 25 έως 30 (εφόσον έχουν ενεργή τη σημαία "collided").

Για παράδειγμα, αν πρέπει να ελέγχουμε μόνο 6 εχθρούς, προρυθμίζοντας την εντολή να ελέγχει μόνο από το 25 και μετά, μπορούμε να εξοικονομήσουμε έως και 2,5ms σε κάθε εκτέλεση. Αυτό γίνεται ιδιαίτερα σημαντικό σε παιχνίδια όπου ο χαρακτήρας μπορεί να πυροβολήσει, αφού σε κάθε κύκλο παιχνιδιού θα πρέπει να ελέγχεται τουλάχιστον η σύγκρουση του χαρακτήρα και των βιολών.

Μια άλλη ενδιαφέρουσα βελτιστοποίηση, ικανή να εξοικονομήσει 1,1 χιλιοστά του δευτερολέπτου σε κάθε κλήση, είναι να πείτε στην εντολή να χρησιμοποιεί πάντα την ίδια μεταβλητή BASIC για να αφήσει το αποτέλεσμα της σύγκρουσης. Για να το κάνουμε αυτό, θα το υποδείξουμε χρησιμοποιώντας την 33 ως sprite, η οποία επίσης δεν υπάρχει.

col%=0

|COLSP, 33, @col%, @col%, @COLSP, 33, @col%, @COLSP, 33, @col%

Μόλις εκτελεστούν αυτές οι δύο γραμμές, οι επόμενες κλήσεις της COLSP θα αφήσουν το αποτέλεσμα στη μεταβλητή col, χωρίς να χρειάζεται να το υποδείξετε, για παράδειγμα:

|COLSP, 23

Τέλος, μπορείτε να ρυθμίσετε την ευαισθησία της εντολής COLSP, αποφασίζοντας αν η επικάλυψη μεταξύ των sprites πρέπει να είναι αρκετά pixel ή μόνο ένα, ώστε να θεωρηθεί ότι έχει συμβεί σύγκρουση.

Αυτό μπορεί να γίνει ορίζοντας τον απαιτούμενο αριθμό επικαλυπτόμενων εικονοστοιχείων τόσο στην κατεύθυνση Y όσο και στην κατεύθυνση X, χρησιμοποιώντας την εντολή COLSP και καθορίζοντας το sprite 34 (το οποίο δεν υπάρχει).

|COLSP, 34, dy, dx

Οι προεπιλεγμένες τιμές για τα dy και dx είναι 2 και 1 αντίστοιχα. Σημειώστε ότι στην κατεύθυνση y θεωρούνται εικονοστοιχεία, αλλά στην κατεύθυνση x θεωρούνται bytes (ένα byte είναι δύο εικονοστοιχεία στη λειτουργία 0).

Για μια ανίχνευση με ελάχιστη επικάλυψη (ένα pixel κάθετα ή/και ένα byte οριζόντια) πρέπει να κάνετε:

|COLSP, 34, 0, 0, 0

21.1.8 |COLSPALL

Χρήση:

Για να ρυθμίσετε τις παραμέτρους:

|COLSPALL,@collider%, @collider%, @collider%, @collider%,
@collider%, @collider%, @collider%.

Για να ελέγξετε για συγκρούσεις

|COLSPALL
|COLSPALL, <αρχικός επιταχυντής>.

Αυτή η συνάρτηση ελέγχει ποιος έχει συγκρουστεί (μεταξύ της ομάδας των sprites που έχουν το collider flag του status byte σε "1") και με ποιον έχει συγκρουστεί (μεταξύ της ομάδας των sprites που έχουν το collider flag του status byte σε "1").

Αυτό είναι ένα ιδιαίτερα συνιστώμενο χαρακτηριστικό όταν πρέπει να χειριστείτε συγκρούσεις του χαρακτήρα σας και πολλαπλές βιολές, καθώς εξοικονομεί κλήσεις του |COLSP και επομένως επιταχύνει το παιχνίδι σας.

Σημαντικό: οι συγκρουστήρες (bit κατάστασης 5) ελέγχονται από το 31 έως το 0. Για κάθε συγκρουστήρα, οι συγκρουστήρες (bit κατάστασης 1) ελέγχονται επίσης από το 31 έως το 0.

Σε περίπτωση που το COLSPALL καλείται με μία μόνο παράμετρο,

|COLSPALL, <αρχικός επιταχυντής>.

Οι συγκρουστές θα σαρώνονται από τον υποδεικνυόμενο συγκρουστή -1 έως το sprite

μηδέν, με φθίνουσα σειρά. Με αυτόν τον τρόπο, αν χρειάζεται να ανιχνεύσετε περισσότερες από μία συγκρούσεις ανά κύκλο του

παιχνίδι, μπορείτε να το κάνετε αυτό καλώντας διαδοχικά **COLSPALL**, <collider> μέχρι η μεταβλητή collider να πάρει την τιμή 32

Παράδειγμα:

| **COLSPALL, 7 : αναζητήσεις rem για συγκρούσεις από τον επιταχυντή 6**

21.1.9 |LAYOUT

Χρήση:

| LAYOUT, <y>, <x>, <@string\$>, <@string\$>, <@string\$>, <@string\$>, <@string\$>, <@string\$>, <@string\$>.

Παράδειγμα:

string\$ = "XYZZZZ ZZ"
| LAYOUT, 0,1, @string\$

Σημειώστε ότι η χρήση του |LAYOUT, 0,1, "XYZZZZZZZZ ZZ" θα ήταν λανθασμένη σε ένα CPC464, αν και λειτουργεί σε ένα CPC6128. Επίσης, στο CPC6128 μπορείτε να παραλείψετε τη χρήση του "@" αλλά στο CPC464 είναι υποχρεωτική.

Αυτή η ρουτίνα εκτυπώνει μια σειρά από sprites για τη δημιουργία της διάταξης ή του "λαβύρινθου" για κάθε οθόνη. Εκτός από τη σχεδίαση του λαβύρινθου ή οποιωνδήποτε γραφικών στην οθόνη που κατασκευάζονται με μικρά sprites 8x8, μπορείτε επίσης να ανιχνεύσετε τις συγκρούσεις ενός sprite με τη διάταξη, χρησιμοποιώντας την εντολή |COLAY.

Τα sprites που πρόκειται να εκτυπωθούν ορίζονται με μια συμβολοσειρά, οι χαρακτήρες της οποίας (32 δυνατοί) αντιπροσωπεύουν ένα από τα sprites που ακολουθούν αυτόν τον απλό κανόνα, όπου η μόνη εξαίρεση είναι το κενό διάστημα που αντιπροσωπεύει την απουσία ενός sprite.

Χαρακτήρας	Id Sprite	Κωδικός ASCII
" "	KANENAΣ	
",	0	59
"<"	1	
"=="		
">"		
"?"		63
"@"	5	
"A"		65
"B"		
"C"	8	67
"D"		
"E"	10	69
"F"		70
"G"		71
"H"		
"I"		

"J"		
"K"		75
"L"		
"M"		
"N"		78
"O"		79
"P"	21	80
"Q"		81
"R"	23	82
"S"		
"T"	25	84
"U"	26	85
"V"		86
"W"		87
"X"	29	88
"Y"	30	
"Z"	31	90

Πίνακας 6 Αντιστοίχιση χαρακτήρων και Sprite για την εντολή |AYOUT

ΣΗΜΑΝΤΙΚΟ: Μετά την εκτύπωση της διάταξης μπορείτε να αλλάξετε τα sprites σε χαρακτήρες, ώστε να εξακολουθείτε να έχετε τα 32 sprites.

Οι συντεταγμένες y,x μεταφέρονται σε μορφή χαρακτήρων. Η βιβλιοθήκη διατηρεί εσωτερικά έναν χάρτη χαρακτήρων 20x25, οπότε οι συντεταγμένες παίρνουν τις ακόλουθες τιμές:

Το y παίρνει τιμές
[0,24] Το x παίρνει
τιμές [0,19].

Τα sprites που πρόκειται να εκτυπωθούν πρέπει να έχουν διαστάσεις 8x8 pixels. Πρόκειται για "τούβλα", που ονομάζονται επίσης "πλακίδια", και χρησιμοποιούνται συχνά με τον ίδιο τρόπο.

Αν χρησιμοποιείτε άλλα μεγέθη sprite, αυτή η λειτουργία δεν θα λειτουργήσει καλά. Θα εκτυπώσει πράγματι τα sprites, αλλά αν ένα sprite είναι μεγάλο, θα πρέπει να τοποθετήσετε κενά για να το χωρέσετε.

Η βιβλιοθήκη διατηρεί έναν εσωτερικό χάρτη διάταξης και αυτή η συνάρτηση ενημερώνει τα δεδομένα του εσωτερικού χάρτη διάταξης ώστε να είναι δυνατός ο εντοπισμός συγκρούσεων. Αυτός ο χάρτης είναι ένας πίνακας 20x25 χαρακτήρων, όπου κάθε χαρακτήρας αντιστοιχεί σε ένα sprite.

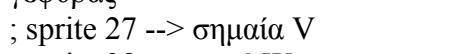
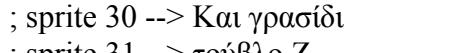
Το @string είναι μια μεταβλητή συμβολοσειράς. Δεν μπορείτε να περάσετε τη συμβολοσειρά απευθείας, αν και στο CPC6128 η παράμετρος που περνάει το επιτρέπει, αλλά θα ήταν ασύμβατο με το CPC464.

Προφυλάξεις:

Η συνάρτηση δεν επικυρώνει τη συμβολοσειρά που της δίνετε. Αν περιέχει πεζά γράμματα ή οποιονδήποτε άλλο χαρακτήρα διαφορετικό από τους επιτρεπόμενους, μπορεί να προκαλέσει ανεπιθύμητα αποτελέσματα, όπως επανεκκίνηση ή κατάρρευση του υπολογιστή. Δεν μπορεί επίσης να είναι κενή συμβολοσειρά!

Τα όρια που έχουν οριστεί με το SETLIMITS θα σας επιτρέψουν να εκτυπώσετε όπου θέλετε. Αν αργότερα θέλετε να κόψετε το αποκορύφωμα σε μια μικρότερη περιοχή, μπορείτε να καλέσετε ξανά το SETLIMITS όταν εκτυπωθεί ολόκληρη η διάταξη.

Παράδειγμα:

2070 SETLIMITS,0,80,0,200 2090 c\$(1)= " PP PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP " 2100 c\$(2)= "PU 2110 c\$(3)= "P 2120 c\$(4)= "P	Αυτό το παράδειγμα χρησιμοποιεί διάφορα τούβλα που έχουν προηγουμένως δημιουργηθεί με το εργαλείο "SPEDIT". ; sprite 20 --> O bush
2130 c\$(5)= "P TPPPPPPPPU TPPPPPPPPPPPPPPPPPPPPPP PPP"	; sprite 21 --> P rock ; sprite 22 --> σύννεφο Q
2140 c\$(6)= "P TP" 2150 c\$(7)= "P P" 2160 c\$(8)= "P P" 2170 c\$(9)= "P	; sprite 23 --> R νερό ; sprite 24 --> παράθυρο S ; sprite 25 --> T αγίδα της δεξιάς γέφυρας ; sprite 26 --> U αγίδα της αριστερής γέφυρας
YYYY YYYYYYYYYYYYYYYYYY 2190 c\$(10)="P P"	; sprite 27 --> σημαία V ; sprite 28 --> φυτό W
PPU TPPPPPPPPPPPPPPPP P"	; sprite 29 --> πύργος X spike ; sprite 30 --> Kai γρασίδι ; sprite 31 --> τούβλο Z
2220 c\$(14)="YYYYYYYYYYYYYYYYYYYYYYYYYYYYYP YYYYYYYYYY" 2230 c\$(15)="RRRRRRRRRRRRRRRRRRRRRRRRRRR". RRRRRRRRRR RR"	
2240 c\$(16)="PPPPPPPPPPPPPPPPPPPPPPPPPP PPPPPPPPPP PPPPPPP"	
2250 c\$(17)="PU TP PU TP" 2260 c\$(18)="P T U P" 2270 c\$(19)="P P" 2271 c\$(20)="P P" 2272 c\$(21)="P W P" 2273 c\$(22)="PP W PP" 2274 c\$(23)="PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP PPPPPPPPPPPPPPPPPPPPPPPPPPPP".	
2280 for i=0 έως 23 2281 LAYOUT,i,0,@c\$(i) 2282 επόμενο	

21.1.10 |LOCATESP

Αυτή η εντολή αλλάζει τις συντεταγμένες ενός sprite στον πίνακα χαρακτηριστικών sprite.

Χρήση

| LOCATESP, <αριθμός θέσης>, <y>, <x>

Παράδειγμα

|LOCATESP,0,10,20

Μια εναλλακτική λύση αυτής της εντολής, αν θέλουμε να αλλάξουμε μόνο μία συντεταγμένη, είναι να χρησιμοποιήσουμε την εντολή POKE της BASIC, εισάγοντας την τιμή που θέλουμε στη διεύθυνση μνήμης που καταλαμβάνει η συντεταγμένη X ή Y. Αν θέλουμε να εισάγουμε μια αρνητική συντεταγμένη, η εντολή |POKE είναι απαραίτητη, καθώς θα ήταν παράνομη με την εντολή BASIC POKE.

Η εντολή |LOCATE δεν εκτυπώνει το sprite, απλώς το τοποθετεί για την εκτύπωση.

21.1.11 |MAP2SP

Αυτή η συνάρτηση διασχίζει τον παγκόσμιο χάρτη που περιγράφεται στο αρχείο map_table.asm και μετατρέπει τα στοιχεία του χάρτη που μπορεί να εισέρχονται μερικώς ή πλήρως στην οθόνη σε sprites.

Χρήση

|MAP2SP, <y>, <x>

|MAP2SP, <status>, <status>, <status>, <status>, <status>, <status>, <status>, <status>, <status>.

Παράδειγμα

|MAP2SP, 1500, 2500

Τα sprites που δημιουργούνται από το MAP2SP δημιουργούνται από προεπιλογή με κατάσταση 3, δηλαδή με ενεργή τη σημαία εκτύπωσης (**|PRINTSPALL** το εκτυπώνει) και με ενεργή τη σημαία σύγκρουσης (**|COLSP** θα συγκρουστεί με αυτό). Αν θέλετε τα sprites να δημιουργηθούν με άλλη κατάσταση, απλά καλείτε την εντολή **|MAP2SP** μία φορά με μία μόνο παράμετρο που υποδεικνύει την κατάσταση με την οποία πρέπει να δημιουργηθούν τα sprites.

Αν κατά τύχη **το |MAP2SP** συναντήσει περισσότερα από 32 αντικείμενα προς μετάφραση σε sprites, θα αγνοήσει αυτά που υπερβαίνουν τα 32.

|MAP2SP, <status>, <status>, <status>, <status>, <status>, <status>, <status>, <status>.

|Αυτή η ρύθμιση ρυθμίζει την εντολή MAP2SP ώστε να εκτυπώνεται αλλά να μην μπορεί να συγκρουστεί.

ΣΗΜΑΝΤΙΚΟ: Στον παγκόσμιο χάρτη μπορείτε να συνδυάσετε κανονικές εικόνες με "εικόνες φόντου" (ενότητα 8.5). Οι εικόνες φόντου έχουν πάντα διαφάνεια. Η σημαία διαφάνειας που χρησιμοποιείτε στο **|MAP2SP, <status>** θα ισχύει μόνο για κανονικές εικόνες.

Οι παράμετροι **<y>, <x>** της συνάρτησης είναι η κινούμενη αφετηρία από την οποία ο κόσμος εμφανίζεται στην οθόνη. Υπάρχουν τρεις άλλες παράμετροι που βρίσκονται στον πίνακα **MAP_TABLE**, τον πίνακα από τον οποίο ορίζεται ο κόσμος. Αυτές οι παράμετροι είναι το μέγιστο ύψος, το μέγιστο πλάτος (σε αρνητική τιμή) και ο αριθμός των στοιχείων του κόσμου (μέγιστο 82).

Κάθε στοιχείο είναι μια πλειάδα 3 παραμέτρων, της οποίας προηγείται το μνημονικό "DW":

DW Y, X, <εικόνα>

ΠΙΝΑΚΑΣ ΧΑΡΤΗΣ

;

3 παράμετροι πριν από τον κατάλογο των "στοιχείων χάρτη".

dw 50 ; μέγιστο ύψος ενός sprite σε περίπτωση που περάσει από την κορυφή και ένα μέρος του πρέπει να ζωγραφιστεί.

dw -40 ; μέγιστο πλάτος ενός sprite σε περίπτωση αριστερόστροφου sprite (αρνητικός αριθμός)

db 64 ; αριθμός των στοιχείων χάρτη που πρέπει να ληφθούν υπόψη. το πολύ να είναι 82

και από εδώ ξεκινούν τα στοιχεία dw

100,10,HOUSE; 1

dw 50,-10,CACTUS;2

dw 210,0,HOUSE;3

dw 200,20,CACTUS;4

dw 100,40,HOUSE;5

dw 160,60,HOUSE;6

dw 70,70,HOUSE;7

dw 175,40,CACTUS;8

dw 10,50,HOUSE;9

dw

250,50,HOUSE;10

dw

260,70,HOUSE;11

dw

260,70,HOUSE;11

dw 290,60,CACTUS;12

dw 180,90,HOUSE;13

dw 60,100,HOUSE;14

dw 60,100,HOUSE;14

...

21.1.12 |MOVER

Αυτή η εντολή μετακινεί ένα sprite σχετικά, δηλαδή προσθέτοντας σχετικές ποσότητες στις συντεταγμένες του.

Χρήση:

| MOVER,<αριθμός θέσης>, <dy>, <dx>, <dx>, <dx>, <dx>, <dx>, <dx>, <dx>, <dx>, <dx>, <dx>.

Παράδειγμα:

| MOVER,0,1,-1

Το παράδειγμα μετακινεί ταυτόχρονα το sprite 0 προς τα κάτω και προς τα δεξιά. Το sprite δεν χρειάζεται να έχει ενεργοποιημένη τη σημαία σχετικής κίνησης.

Υπάρχει ένας τρόπος να χρησιμοποιήσετε το **| MOVER** χωρίς να προσδιορίσετε είτε το "dy" είτε το "dx". Για να το κάνουμε αυτό, θα καθορίσουμε το sprite 32, το οποίο δεν υπάρχει, και θα βάλουμε ως παραμέτρους τις διεύθυνσεις μνήμης των μεταβλητών που θέλουμε να χρησιμοποιήσουμε για να αποθηκεύσουμε τόσο το "dy" όσο και το "dx".

Η διεύθυνση μνήμης μιας μεταβλητής λαμβάνεται με την απλή προσάρτηση του συμβόλου "@".

Παράδειγμα:

dy%= 5

dx%= 2

| MOVER,32, @dy, @dx

Από αυτή τη στιγμή, θα μπορούμε να χρησιμοποιούμε:

| MOVER, <id>

Και με αυτό το sprite "id" θα κινηθεί όπως υποδεικνύεται από τις μεταβλητές dy, dx. Αυτός ο μηχανισμός λειτουργεί επίσης με **| MOVERALL**

21.1.13 |MOVERALL

Αυτή η εντολή μετακινεί σχετικά όλα τα sprites που έχουν ενεργοποιημένη τη σημαία σχετικής κίνησης.

Χρήση

| MOVERALL, <dy>, <dx>, <dx>, <dx>, <dx>, <dx>, <dx>, <dx>, <dx>.

Παράδειγμα

| MOVERALL,2,1

Το παράδειγμα μετακινεί όλα τα sprites με σημαία σχετικής κίνησης προς τα κάτω (2 γραμμές) και 1 byte προς τα δεξιά.

Εάν δεν καθοριστούν παράμετροι, θα χρησιμοποιηθούν οι μεταβλητές που καθορίζονται στην κλήση MOVER με το sprite 32, δηλαδή.

| MOVER,32, @dy, @dx

| MOVERALL

Ισοδύναμο με **|MOVERALL**, dy, dx

Αυτή η "προηγμένη" χρήση της εντολής αποφεύγει τη μετάδοση παραμέτρων σε κάθε

κλήση και είναι επομένως ταχύτερη, πράγμα που είναι απαραίτητο στα προγράμματα BASIC μας.

21.1.14 |MUSIC

Αυτή η εντολή επιτρέπει την έναρξη αναπαραγωγής μιας μελωδίας Χρήση:

|MUSIC,<flag_channel_C>,<flag_repeat>,<melody_number>,<speed>.
|MUSIC,<flag_repetition>,<melody_number>,<speed>.

|MUSIC : ' χωρίς παραμέτρους η μουσική τελειώνει την αναπαραγωγή

Η σημαία C-channel με τιμή 1 επιτρέπει στο τρίτο κανάλι ήχου να μείνει ελεύθερο, ώστε να μπορεί να χρησιμοποιηθεί για την παραγωγή ηχητικών εφέ (σκανδαλισμοί κ.λπ.) με την εντολή

SOUND 4,<note>,<duration>, ...

Σημειώστε ότι πρέπει να χρησιμοποιήσετε το κανάλι 4, καθώς στη BASIC τα κανάλια πληκτρολογούνται ως A=1, B=2, C=4.

Αν η σημαία καναλιού παραλειφθεί ή μηδενιστεί, τότε η μουσική θα χρησιμοποιεί και τα 3 κανάλια και δεν θα μπορείτε να χρησιμοποιήσετε ταυτόχρονα την εντολή SOUND (μπορείτε, αλλά με περίεργα αποτελέσματα).

Η σημαία επανάληψης πρέπει να είναι μηδέν για να επαναλαμβάνεται η μουσική σε επανάληψη. Αν θέλετε η μουσική να αναπαράγεται μόνο μία φορά, θα πρέπει να χρησιμοποιήσετε την τιμή 1.

Ο αριθμός μελωδίας πρέπει να είναι μεταξύ 0 και 7.

Η "κανονική" ταχύτητα είναι 6. Αν χρησιμοποιήσουμε μεγαλύτερο αριθμό θα παίξει πιο αργά και αν ο αριθμός είναι μικρότερος θα παίξει πιο γρήγορα.

Η εντολή |MUSIC που καλείται χωρίς παραμέτρους απενεργοποιεί τη διακοπή της μουσικής και σταματά την αναπαραγωγή οποιασδήποτε μελωδίας.

Παραδείγματα:

|MUSIC,0,0,0,0,0,6
|MUSIC,1,0,0,0,0,6
|MUSIC,0,0,6
|MUSIC

Εσωτερικά, αυτό που κάνει η εντολή είναι να εγκαταστήσει μια διακοπή που ενεργοποιείται 300 φορές ανά δευτερόλεπτο. Αν ορίσουμε ταχύτητα 6, μία από τις 6 φορές που ενεργοποιείται, εκτελείται η λειτουργία αναπαραγωγής μουσικής.

Επειδή βασίζεται σε διακοπές, πρέπει να εκτελείται ένα πρόγραμμα για να παίξει η μουσική, επειδή όσο ο διερμηνέας BASIC περιμένει εντολές, οι διακοπές αυτές δεν είναι ενεργοποιημένες. Αν απλά εκτελέσετε την εντολή |MUSIC, δεν θα ακούσετε τίποτα, αλλά αν την εκτελέσετε μέσα σε ένα πρόγραμμα όπως αυτό που φαίνεται παρακάτω, η μουσική θα παίξει.

10 |MUSIC,0,0,0,0,5

20 goto 20: ' άπειρος βρόχος. Κατά την εκτέλεση, η μουσική παίζει

21.1.15 |PEEK

Αυτή η εντολή διαβάζει μια τιμή δεδομένων 16 bit από μια δεδομένη διεύθυνση

μνήμης. Προορίζεται για την αναζήτηση των συντεταγμένων των sprites που κινούνται με αυτόματη ή σχετική κίνηση.

Χρήση
|PEEK,<διεύθυνση>, @data%.

Παράδειγμα
data%=0
|PEEK, 27001, @dato%, @dato%, @dato%, @dato%, @dato%, @pEEK

Εάν οι συντεταγμένες είναι μόνο θετικές και μικρότερες από 255, μπορείτε να χρησιμοποιήσετε την εντολή PEEK της BASIC, καθώς είναι κάπως ταχύτερη.

21.1.16 |POKE

Αυτή η εντολή εισάγει ένα δεδομένο 16 bit (θετικό ή αρνητικό) σε μια διεύθυνση μνήμης. Προορίζεται για την τροποποίηση των συντεταγμένων του sprite, καθώς η εντολή POKE δεν μπορεί να χειριστεί αρνητικές συντεταγμένες ή συντεταγμένες μεγαλύτερες από 255, καθώς η POKE λειτουργεί με bytes, ενώ η |POKE είναι μια εντολή 16bit.

Χρήση:
|POKE, <διεύθυνση>, <τιμή>.

Παράδειγμα:
|POKE, 27003, 23

Αυτό το παράδειγμα τοποθετεί την τιμή 23 στη συντεταγμένη x του sprite 0. Πρόκειται για μια πολύ γρήγορη συνάρτηση, αν και αν πρόκειται να χειριστείτε μόνο θετικές συντεταγμένες, είναι προτιμότερο να χρησιμοποιήσετε την POKE, καθώς είναι ακόμη πιο γρήγορη.

21.1.17 |PRINTAT

|PRINTAT μπορεί να εκτυπώσει μια σειρά χαρακτήρων χρησιμοποιώντας ένα νέο, μικρότερο σύνολο χαρακτήρων (τους ονομάζω "μίνι-χαρακτήρες"). Αυτή η νέα εντολή σας επιτρέπει να χρησιμοποιήσετε το μηχανισμό διαφάνειας των sprites, ώστε να μπορείτε να εκτυπώνετε χαρακτήρες με σεβασμό στο φόντο. Λειτουργεί ως εξής:

Χρήση:
|PRINTAT,<σημαία διαφάνειας>, y,x,@string

Παράδειγμα:
cad\$= "Hello".
|PRINTAT,0,100,10, @cad\$

Η εντολή |PRINTAT εκτυπώνει συμβολοσειρές χαρακτήρων και όχι αριθμητικές μεταβλητές, οπότε αν θέλετε να εκτυπώσετε έναν αριθμό (για παράδειγμα, τους πόντους στον πίνακα αποτελεσμάτων στο βιντεοπαιχνίδι σας) πρέπει να το κάνετε:

```
points=points+1
cad$= str$(points)
|PRINTAT,0,100,10, @cad$
```

Η εντολή |PRINTAT δεν επηρεάζεται από τα όρια αποκοπής που έχουν οριστεί με την εντολή

|SETLIMITS. Αυτό είναι το πιο λογικό, δεδομένου ότι συνήθως θα χρησιμοποιήσετε το PRINTAT για να εκτυπώσετε αποτελέσματα στους δείκτες σας, οι οποίοι θα βρίσκονται εκτός της περιοχής που ορίζεται από το |SETLIMITS.

Σε αντίθεση με την εντολή PRINT της BASIC, η εντολή |PRINTAT είναι αρκετά γρήγορη και μπορεί να χρησιμοποιηθεί για να ενημερώνετε συχνά τους δείκτες του παιχνιδιού σας.

Το PRINTAT χρησιμοποιεί ένα επαναπροσδιορισμένο αλφάριθμο, το οποίο μπορεί να περιέχει μια μειωμένη ή διαφορετική έκδοση των "επίσημων" χαρακτήρων της Amstrad. Από προεπιλογή, το 8BP παρέχει ένα μικρό αλφάριθμο που αποτελείται από αριθμούς, κεφαλαία γράμματα και ορισμένα σύμβολα. Είναι το ακόλουθο:

"0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ!: ,."

Δεν θα μπορέσετε να χρησιμοποιήσετε χαρακτήρα που δεν περιλαμβάνεται σε αυτό το σύνολο, όπως πεζά γράμματα. Αν προσπαθήσετε να το κάνετε αυτό, θα εκτυπωθεί ο τελευταίος χαρακτήρας που ορίζεται στη συμβολοσειρά (στην προκειμένη περίπτωση το κενό).

Οι χαρακτήρες αυτού του αλφαριθμού έχουν όλοι το ίδιο μέγεθος: 4 εικονοστοιχεία πλάτος x 5 εικονοστοιχεία ύψος, δηλαδή 2 bytes x 5 γραμμές.

Το προεπιλεγμένο αλφάριθμο δεν περιέχει πεζά γράμματα και πολλά σύμβολα λείπουν, αν και μπορείτε να δημιουργήσετε το δικό σας αλφάριθμο που τα περιέχει.

21.1.18 |PRINTSP

Χρήση:

| PRINTSP, <prite id >, <y >,<x>, <x>.
| PRINTSP, < αναγνωριστικό πτήσης >
| PRINTSP, 32, <αριθμός εικονοστοιχείων φόντου>.

Παράδειγμα:

Εκτύπωση του sprite 23 στις συντεταγμένες y=100, x=40 (ενημέρωση των συντεταγμένων του).

|PRINTSP, 23,100,40

Παράδειγμα:

Εκτυπώνει το sprite 23 στις συντεταγμένες που έχουν ήδη οριστεί στον πίνακα sprite:

|PRINTSP, 23

Εάν καθοριστεί το sprite 32 (το οποίο δεν υπάρχει), τότε χρησιμοποιείται η ακόλουθη παράμετρος για να καθοριστεί ο αριθμός των bits φόντου στη διαφανή εκτύπωση. Εάν χρησιμοποιείται 1 bit, τότε μπορούν να χρησιμοποιηθούν 2 χρώματα φόντου. Εάν καθοριστούν 2, τότε μπορούν να χρησιμοποιηθούν 4 χρώματα φόντου.

Αν καθοριστεί ένα sprite_id <32, τότε η εντολή εκτυπώνει ένα sprite στην οθόνη, και αν καθοριστούν οι συντεταγμένες, τις ενημερώνει επίσης.

Οι συντεταγμένες που εξετάζονται είναι:

- Αριθμός γραμμών κάθετα [-32768..32768]. Οι αντίστοιχες γραμμές στο

- εσωτερικό της οθόνης είναι [0..199].
- Αριθμός bytes σε οριζόντιο επίπεδο [-32768..32768]. Αυτά που αντιστοιχούν στο εσωτερικό της οθόνης είναι [0..79].

Συνήθως, στη λογική ενός βιντεοπαιχνιδιού θα χρησιμοποιήσετε την εντολή **|PRINTSPALL**, καθώς είναι πιο γρήγορο να τα εκτυπώσετε όλα μαζί. Ωστόσο, σε άλλες στιγμές του παιχνιδιού μπορεί να θέλετε να εκτυπώσετε τα sprites ξεχωριστά. Αυτό το παράδειγμα δείχνει το κατέβασμα μιας "κουρτίνας", χρησιμοποιώντας ένα μόνο sprite που επαναλαμβάνεται οριζόντια και καθώς κατεβαίνει "χρωματίζει" την οθόνη με κόκκινο χρώμα, δίνοντας την αίσθηση ότι κατεβαίνει μια κουρτίνα.

```

7089 telon=&8ec0
7090 |setupsp,1,9,curtain
7100 για y=8 έως 168 βήμα 4
7110 για x=12 έως 64 βήμα 4
7111 |PRINTSP,1,y,x
7112 επόμενο
7113 επόμενος

```



Σχ. 119 Παράδειγμα χρήσης του PRINTSP

21.1.19 |PRINTSPALL

Αυτή η ρουτίνα εκτυπώνει όλα τα sprites με το bit0 κατάστασης ρυθμισμένο ταυτόχρονα.

Χρήση:

|PRINTSPALL, <ordenini>, <ordenfin>, <flag anima>, <flag sync>.
|PRINTSPALL, <τύπος παραγγελίας>,

Παράδειγμα:

Με τις ακόλουθες τιμές, η εντολή εκτυπώνει όλα τα sprites με πρώτη κίνηση και χωρίς συγχρονισμό με το sweep και χωρίς ταξινόμηση:

|PRINTSPALL, 0, 0, 0, 0, 1, 0

|PRINTSPALL, 0, 1, 0: rem αν η εντολή ινι παραλείπεται, παίρνει την τελευταία τιμή που δόθηκε, ή μηδέν αν δεν δόθηκε ποτέ

animation flag, μπορεί να οριστεί σε 1 ή 0. Εάν οριστεί 1, τότε πριν από την εκτύπωση των sprites το πλαίσιο αλλάζει στην ακολουθία animation του, εφόσον τα sprites έχουν ορισμένο το bit κατάστασης 3. **ΣΗΜΑΝΤΙΚΟ:** Το animation γίνεται πριν από την εκτύπωση, όχι μετά την εκτύπωση. Αυτό σημαίνει ότι αν έχετε μόλις εκχωρήσει μια ακολουθία κινούμενων σχεδίων, δεν θα δείτε το πρώτο καρέ της ακολουθίας αυτής.

Η **<flag sync>** είναι μια σημαία συγχρονισμού με τη σάρωση οθόνης. Μπορεί να είναι 1 ή 0. Ο συγχρονισμός έχει νόημα μόνο αν μεταγλωττίσετε το πρόγραμμα με ένα μεταγλωττιστή όπως το "Fabacom". Η λογική της BASIC τρέχει αργά και ο συγχρονισμός με τη σάρωση παράγει μικρές πρόσθετες αναμονές σε κάθε κύκλο του παιχνιδιού, οπότε δεν είναι βολικό.

Ως γενικός κανόνας, είναι κατάλληλο μόνο αν το παιχνίδι σας είναι ικανό να παράγει 50fps ανά δευτερόλεπτο, ή με άλλα λόγια, έναν πλήρη κύκλο παιχνιδιού κάθε 20 χιλιοστά του δευτερολέπτου. Εάν μεταγλωττίσετε το παιχνίδι σας με έναν

μεταγλωττιστή όπως το "fabacom", τότε συνιστάται ο συγχρονισμός με τη σάρωση της οθόνης, επειδή σχεδόν σίγουρα θα επιτύχετε αυτά τα 50fps και θα μπορείτε να έχετε έναν πλήρη κύκλο παιχνιδιού κάθε 20 χιλιοστά του δευτερολέπτου.

Αν τις υπερβείτε, το παιχνίδι σας θα παράγει περισσότερα καρέ από όσα μπορεί να εμφανίσει η οθόνη και τότε κάποια δεν θα εμφανίζονται και η κίνηση δεν θα είναι ουμαλή.

Όσο περισσότερα sprites έχετε στην οθόνη εκτύπωσης, τόσο περισσότερο χρόνο θα χρειαστεί η εντολή, αν και είναι πολύ γρήγορη. Υπάρχουν πολλά sprites που μπορεί να εμφανίζονται στην οθόνη, αλλά δεν χρειάζεται να εκτυπωθούν (μπορεί να έχουν απενεργοποιημένο το bit κατάστασης 0), όπως φρούτα, νομίσματα, στοιχεία μπόνους γενικά ή/και χαρακτήρες που δεν κινούνται και δεν έχουν animation. Ακόμη και αν δεν εκτυπώνονται, μπορεί να έχουν το bit σύγκρουσης ρυθμισμένο και έτσι να επηρεάζουν τη ρουτίνα.

|COLSP kai |COLSPALL

Οι παράμετροι τάξης ("ordenini", "ordenfin") υποδεικνύουν το αρχικό και το τελικό sprite που καθορίζουν την ομάδα sprites ταξινομημένων με βάση τη συντεταγμένη "Y" που πρόκειται να εκτυπώσουμε. Για παράδειγμα, αν δώσουμε τις τιμές 0,0 τότε θα εκτυπωθούν διαδοχικά από το sprite 0 έως το sprite 31. Αν δώσουμε τις τιμές 0,8 θα εκτυπωθούν από το 0 έως το 8 διατεταγμένα (9 sprites) και από το 10 έως το 31 διαδοχικά. Εάν ορίσουμε 0,31 όλα τα sprites θα εκτυπωθούν με τη σειρά. Αν ορίσουμε ένα 10,20 τα sprites θα εκτυπωθούν διαδοχικά από το 0 έως το 9, στη συνέχεια θα εκτυπωθούν διατεταγμένα από το 10 έως το 20 και τέλος θα εκτυπωθούν διαδοχικά από το 21 έως το 31.

Η διάταξη είναι πολύ χρήσιμη για την κατασκευή παιχνιδιών τύπου "Renegade" ή "Golden AXE", όπου είναι απαραίτητο να δοθεί ένα εφέ βάθους.

Η εκτύπωση με διατεταγμένο τρόπο είναι πιο δαπανηρή υπολογιστικά από την εκτύπωση με διαδοχική σειρά. Αν έχετε μόνο 5 sprites που πρέπει να ταξινομηθούν, περάστε ένα 4 ως παράμετρο ταξινόμησης, μην περάστε ένα 31. Η ταξινόμηση όλων των sprites διαρκεί περίπου 2,5ms, αλλά αν ταξινομήσετε μόνο 5 μπορείτε να εξοικονομήσετε 2ms. Ίσως έχετε πολλά sprites και δεν αξίζει να ταξινομήσετε κάποια από αυτά, όπως τα πλάνα ή τα sprites που ξέρετε ότι δεν θα επικαλύπτονται.

Η ταξινόμηση του **PRINTSPALL** είναι μερική, δηλαδή μόνο ένα ζεύγος μη ταξινομημένων sprites ταξινομείται σε κάθε κλήση. Μερικές φορές μπορεί να θέλετε η ταξινόμηση να είναι πλήρης για κάθε καρέ. Δηλαδή, να μην ταξινομείτε ένα ζεύγος sprites σε κάθε κλήση του **PRINTSPALL**, αλλά να είστε σίγουροι ότι είναι όλα ταξινομημένα. Η βιβλιοθήκη 8BP σας επιτρέπει να το κάνετε αυτό μέσω των τεσσάρων τρόπων ταξινόμησης, τους οποίους μπορείτε να ορίσετε με την κλήση της εντολής **PRINTSPALL** με μία μόνο παράμετρο (απλά εκτελέστε την μία φορά για να ορίσετε τον τρόπο ταξινόμησης):

PRINTSPALL,0 : μερική ταξινόμηση με χρήση Y_{min}
PRINTSPALL,1 : πλήρης ταξινόμηση με χρήση Y_{min}
PRINTSPALL,2 : μερική ταξινόμηση με χρήση Y_{max}
PRINTSPALL,3 : πλήρως ταξινόμηση με χρήση Y_{max}

Η ταξινόμηση με τη χρήση Ymax βασίζεται στη μεγαλύτερη συντεταγμένη Y των sprites, δηλαδή στο σημείο που βρίσκονται τα πόδια τους και όχι τα κεφάλια τους. Αν τα sprites έχουν το ίδιο μέγεθος, η ταξινόμηση με βάση την Ymin μπορεί να λειτουργήσει, αλλά αν τα sprites έχουν διαφορετικό ύψος, μπορεί να θέλετε να ταξινομήσετε σύμφωνα με το πού βρίσκονται τα πόδια κάθε χαρακτήρα, και γι' αυτό θα πρέπει να χρησιμοποιήσετε τη λειτουργία ταξινόμησης 2 ή 3.

Οι λειτουργίες ταξινόμησης Ymax είναι πιο αργές, περίπου 0,128 ms ανά sprite, οπότε χρησιμοποιήστε τις όταν τις χρειάζεστε πραγματικά.

Η πλήρης ταξινόμηση καταναλώνει πολύ λίγο περισσότερο από τη μερική ταξινόμηση (περίπου 0,3ms). Αυτό οφείλεται στο γεγονός ότι τα sprites δεν είναι σχεδόν ποτέ ανακατεμένα από το ένα καρέ στο άλλο, αλλά ακόμη και αυτά τα 0,3ms αξίζει να τα εξοικονομήσετε αν είναι δυνατόν.

Υπάρχει μια πολύ ενδιαφέρουσα συμπεριφορά αυτής της συνάρτησης που εξοικονομεί 1ms στην εκτέλεσή της. Συνίσταται στην κλήση της με παραμέτρους μία φορά και στην κλήση της χωρίς παραμέτρους τις επόμενες φορές. Σε αυτή την περίπτωση, θα θεωρηθεί ότι, ακόμη και αν δεν έχουν περάσει παράμετροι, οι τιμές τους είναι ίσες με τις τελευταίες που έχουν περάσει. Με αυτόν τον τρόπο ο αναλυτής εργάζεται λιγότερο και μειώνεται ο χρόνος εκτέλεσης.

21.1.20 |RINK

Αυτή η εντολή σας επιτρέπει να εκτελέσετε μια κίνηση με μελάνια. Χρήση:

| RINK, <initial_ink>, <colour1>, <colour2>, , <colourN>, <colourN>, <colourN>, <colourN>, <colourN>, <colourN>, <colourN>, <colourN>, <colourN>, <βήμα>

Η RINK περιστρέφει ένα σύνολο μελανιών ξεκινώντας από το αρχικό μελάνι N μελάνια (οποιοσδήποτε αριθμός μελανιών), σύμφωνα με το μέγεθος του χρωματικού μοτίβου που έχει οριστεί.

Η ταχύτητα περιστροφής μπορεί να ελεγχθεί από την παράμετρο step, η οποία υποδεικνύει τον αριθμό των αλμάτων χρώματος που κάνει κάθε μελάνι σε μία κλήση.

Σύσταση: λόγω της χρήσης διακοπών, **η εντολή |RINK** προκαλεί "τραύλισμα" σε ορισμένες περιπτώσεις όταν χρησιμοποιείται ταυτόχρονα με την εντολή **|MUSIC** στην ταχύτητα 6. Σε περίπτωση που θέλετε να χρησιμοποιήσετε και τις δύο ταυτόχρονα χωρίς παρεμβολές, χρησιμοποιήστε άλλη ταχύτητα για τη μουσική (μπορείτε να χρησιμοποιήσετε την ταχύτητα 5 ή 7, και οι δύο θα λειτουργήσουν άψογα).

Παραδείγματα:

Αυτή η εντολή ορίζει ένα μοτίβο 4 κόκκινων (χρώμα =3) και 4 κίτρινων (χρώμα =24) που θα περιστρέφονται από το μελάνι 8 στο μελάνι 15.

| RINK, 8, 3, 3, 3, 3 , 3, 3, 3, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24

Αυτή η εντολή ορίζει ένα μοτίβο από 2 λευκά (χρώμα =26) και 2 γκρι (χρώμα =13) που θα περιστραφούν από το μελάνι 3 στο μελάνι 6.

| RINK, 3, 26, 26, 13, 13, 13, 13

Περιστρέψτε ένα χρώμα του μοτίβου όλα τα μελάνια

| RINK, 1

Σε ένα μοτίβο 8 χρωμάτων, η ακόλουθη εντολή αφήνει τα πάντα ως έχουν

| RINK, 8

Αυτή η εντολή δεν θα έκανε τίποτα άλλο εκτός από το να αναγκάσει τα μελάνια να

υιοθετήσουν το χρώμα του σχεδίου.

| RINK, O

|ROUTEALL

Αυτή η εντολή σας επιτρέπει να δρομολογήσετε όλα τα sprites που έχουν ενεργή τη σημαία δρομολόγησης στο byte κατάστασης μέσω της διαδρομής που τους έχει ανατεθεί (παράμετρος 15 της |SETUPSP).

Χρήση:

|ROUTEALL

Δεν έχει παραμέτρους, οπότε είναι πολύ εύκολο να την καλέσετε. Αυτό που κάνει εσωτερικά αυτή η εντολή είναι να κρατάει έναν αριθμό βημάτων για το τμήμα που τρέχει κάθε sprite, έτσι ώστε αν το τμήμα τελειώσει, να αλλάξει την ταχύτητα του sprite.

Η εντολή δεν τροποποιεί τις συντεταγμένες των sprites, οπότε πρέπει να μετακινηθούν με την εντολή |AUTOALL και να εκτυπωθούν (και να εμψυχωθούν) με την εντολή |PRINTSPALL. Γι' αυτό το λόγο έχετε μια προαιρετική παράμετρο στην |AUTOALL, έτσι ώστε η |AUTOALL,1 να καλεί εσωτερικά την |AUTOALL,1.

|ROUTEALL πριν μετακινήσετε το sprite, γλιτώνοντάς σας από μια κλήση της BASIC που θα διαρκεί πάντα ένα πολύτιμο χλιοστό του δευτερολέπτου.

Οι διαδρομές ορίζονται στο αρχείο routes_yourgame.asm.

ΟΡΙΣΜΟΣ ΚΑΘΕ ΔΙΑΔΡΟΜΗΣ

=====

ROUTE0- ένας κύκλος

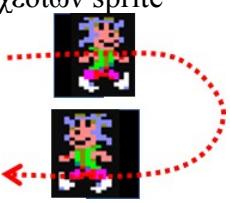
```
db 5,2,0
db 5,2,-1
db 5,0,-1
db 5,-2,-1
db 5,-2,0
db 5,-2,1
db 5,0,1
db 5,2,1
db 0
```

Το τελευταίο τμήμα είναι μηδέν, υποδεικνύοντας ότι το μονοπάτι έχει τελειώσει και το sprite πρέπει να ξεκινήσει από την αρχή. Βεβαιωθείτε ότι ο αριθμός των βημάτων σε κάθε τμήμα δεν υπερβαίνει τα 250 και ότι τόσο το Vy όσο και το Vx είναι μεταξύ -127 και 127.

Για να εκχωρήσετε μια διαδρομή σε ένα sprite πρέπει να χρησιμοποιήσετε την εντολή SETUPSP καθορίζοντας την παράμετρο 15. Το ακόλουθο παράδειγμα συσχετίζει τη διαδρομή 3 με το sprite 31

|SETUPSP, 31, 15, 3

Υπάρχουν 4 λειτουργίες που μπορείτε να χρησιμοποιήσετε στη μέση (όχι στο τέλος) οποιασδήποτε διαδρομής:

Κωδικός διαφυγής (πεδίο "αριθμός βημάτων")	Περιγραφή	Παράδειγμα
255	Αλλαγή της κατάστασης του sprite.	DB 255, 3, 0 Η κατάσταση πηγαίνει στην τιμή 3. Το μηδέν στο τέλος είναι ένα συμπλήρωμα.
254	Αλλαγή της ακολουθίας κινούμενων σχεδίων sprite  Αφού αλλάξετε την ακολουθία, αν θέλετε να αλλάξει και η εικόνα, πρέπει να χρησιμοποιήσετε τον κωδικό 251	DB 254, 10, 0 Η ακολουθία 10 συνδέεται. Το μηδέν είναι ένα γέμισμα. Αν η ακολουθία που έχει εικωρηθεί είναι αυτή που έχει ήδη το sprite, τότε δεν υπάρχει πρόβλημα (το id του πλαισίου δεν μηδενίζεται). Σε περίπτωση που θέλετε να επαναφέρετε το αναγνωριστικό πλαισίου, η τρίτη παράμετρος πρέπει να είναι 1, π.χ. DB 254, 10, 1.
253	Αλλαγή εικόνας 	DB 253 DW new_img Συνδέεται η εικόνα "new_img", η οποία πρέπει να είναι μια διεύθυνση μνήμης.
252	Αλλαγή διαδρομής	DB 252, 2, 0 Η διαδρομή 2 συνδέεται
251	Πηγαίνετε στο επόμενο πλαίσιο από το animation. 	DB 251, 0, 0, 0 Το Sprite είναι κινούμενο. Τα δύο μηδενικά είναι fillers

ΣΗΜΑΝΤΙΚΟ: προσέξτε πολύ να γράφετε DB και DW εκεί που πρέπει να χρησιμοποιηθούν, δηλαδή, για παράδειγμα, αν αλλάζετε εικόνες, θα πρέπει να προηγείται της εικόνας το DW και όχι το DB. Εάν κάνετε ένα τέτοιο λάθος, η διαδρομή σας δεν θα λειτουργήσει.

ΣΗΜΑΝΤΙΚΟ: Μια διαδρομή μπορεί να έχει μήκος το πολύ 255 bytes και ένα τμήμα έχει μήκος 3 bytes, οπότε μια διαδρομή μπορεί να έχει μήκος το πολύ 84 τμήματα. Μπορεί να χρειαστεί να δημιουργήσετε μια ακόμα μεγαλύτερη διαδρομή και σε αυτή την περίπτωση μπορείτε να το κάνετε με τη συνένωση του τέλους μιας διαδρομής με

μια αλλαγή διαδρομής σε μια άλλη διαδρομή (κωδικός 252), και μπορείτε να συνθέσετε όσες διαδρομές θέλετε.

ΣΗΜΑΝΤΙΚΟ: οι κωδικοί διαφυγής μπορούν να χρησιμοποιηθούν στη μέση μιας διαδρομής, αλλά το τελευταίο τμήμα δεν μπορεί να είναι κωδικός διαφυγής, πρέπει να είναι μια κίνηση, ακόμη και αν είναι ακίνητο, κάτι σαν "DB 1,0,0,0".

Σε αυτές τις περιπτώσεις, το |ROUTEALL θα ερμηνεύσει ότι πρέπει να εξαναγκαστεί μια αλλαγή κατάστασης, ακολουθίας, εικόνας, διαδρομής sprite ή κίνησης και θα εκτελέσει επίσης το επόμενο βήμα. Οι αλλαγές μπορούν να εξαναγκαστούν σε οποιοδήποτε τμήμα της διαδρομής, όχι απαραίτητα στο τέλος, και μπορείτε να εξαναγκάσετε όσες αλλαγές θέλετε.

ROUTE0; μία βολή αριστερά

;-----

```
db 100,0,-1; εκατό βήματα αριστερά με ταχύτητα Vx=-1
db 255,0,0; απενεργοποίηση του sprite με status=0
db 1,0,0 ; μην μετακινήσετε τίποτα σε
αυτό το βήμα db 0
```

ROUTE1; ένα άλμα προς τα

δεξιά db 253

dw SOLDIER_R1_UP

db 1,-5,1

db 2,-4,1

db 2,-3,1

db 2,-2,1

db 2,-1,1

db 253

dw SOLDIER_R1_DOWN

db 1,-5,1; πάνω, ώστε UP και κάτω να

ταριάζουν db 2,1,1

db 2,2,1

db 2,3,1

db 2,4,1

db 1,5,1

db 253

dw SOLDIER_R1

db 1,5,1; κάτω ένα ακόμα

db 255,13,0- νέα κατάσταση, χωρίς σημαία διαδρομής και με

σημαία κίνησης db 254,32,0- ακολουθία μακροεντολών 32

db 1,0,0; quietooo!!!!

db 0

|ROUTEESP

Αυτή η εντολή σας επιτρέπει να δρομολογήσετε ένα μεμονωμένο Sprite που έχει ενεργή τη σημαία δρομολόγησης στο byte κατάστασής του μέσω της διαδρομής που του έχει ανατεθεί (παράμετρος SETUPSP 15).

Χρήση:

|ROUTEESP, <sprited>, <steps>.

|ROUTEESP, <spriteid> : rem σε αυτή την περίπτωση το "βήματα"

Θεωρείται=1 Αυτή η εντολή μετακινεί ένα sprite κατά μήκος της διαδρομής που του έχει οριστεί, με οποιοδήποτε αριθμό βημάτων (έως 255). Η εντολή μετακινεί το sprite μέσω όλων των βημάτων που του υποδεικνύονται, αφήνοντάς το τελικά να βρίσκεται στην ίδια θέση που θα είχαμε εκτελέσει την εντολή |AUTOALL,1 αριθμό φορών ίσο με τον αριθμό των βημάτων.

ΣΗΜΑΝΤΙΚΟ: τα βήματα δεν μπορούν να πάρουν τιμή μεγαλύτερη από 255

21.1.23 |SETLIMITS

Αυτή η εντολή ορίζει τα όρια της περιοχής στην οποία μπορούν να εκτυπωθούν sprites ή αστέρια.

Χρήση:

Παράδειγμα που ορίζει ολόκληρη την οθόνη ως επιτρεπόμενη περιοχή | **SETLIMITS,0,80,0,200**

Εκτός αυτών των ορίων, τα sprites περικόπτονται, έτσι ώστε αν ένα sprite είναι μερικώς μερικώς έξω από το των περιοχή επιτρέπεται, το λειτουργίες |PRINTSP y

|Η PRINTSPALL θα εκτυπώσει μόνο το τμήμα που βρίσκεται εντός της επιτρεπόμενης περιοχής.

21.1.24 |SETUPSP

Αντή η εντολή φορτώνει δεδομένα από ένα sprite στη χρήση SPRITES TABLE:

|SETUPSP, <id_sprite>, <param_number>, <value>.

Παράδειγμα:

|SETUPSP, 3, 7, 2

Επιτρέπει, για παράδειγμα, την ανάθεση μιας νέας ακολουθίας κινούμενων σχεδίων όταν το sprite αλλάζει κατεύθυνση, ή απλά την αλλαγή του καταχωρητή σημαιών κατάστασης.

Με αυτή τη συνάρτηση μπορούμε να αλλάξουμε οποιαδήποτε παράμετρο ενός sprite, εκτός από τα X, Y (που γίνεται με την LOCATE SPRITE).

Μπορούμε να αλλάξουμε μόνο μία παράμετρο κάθε φορά. Η προς αλλαγή παράμετρος καθορίζεται με τον αριθμό `param_number`. Ο αριθμός `param_number` είναι στην πραγματικότητα η σχετική θέση της παραμέτρου στον ΠΙΝΑΚΑ SPRITES TABLE.

Αριθμός Param	Δράση	Πιθανή χρήση των POKE ή POKE ως εναλλακτική λύση
0	αλλάζει την κατάσταση (καταλαμβάνει 1 byte)	NAI
5	αλλάζει το Vy (καταλαμβάνει 1byte, τιμή σε κάθετες γραμμές). Μπορείτε επίσης να αλλάξετε Vx την ίδια στιγμή, αν το προσθέσουμε στο τέλος ως παράμετρο	NAI
	αλλάζει Vx (καταλαμβάνει 1byte, τιμή σε οριζόντια bytes)	NAI

	ακολουθία αλλαγής (καταλαμβάνει 1byte, λαμβάνει τιμές 0..31)	ΟΧΙ , επειδή το SETUPSP επαναφέρει επίσης το frame_id και σας δίνει επίσης εικωρεί τη διεύθυνση της πρώτης εικόνας.
8	αλλαγή frame_id (καταλαμβάνει 1byte, λαμβάνει τιμές 0..7)	ΝΑΙ
	αλλαγή εικόνας dir (καταλαμβάνει 2bytes). Η καθορισμένη εικόνα μπορεί να είναι μία από την αρχική λίστα εικόνων στο αρχείο images_mygame.asm,	Δεν είναι το ίδιο εάν χρησιμοποιείται εικόνα <255. Εάν χρησιμοποιείται διεύθυνση μνήμης, το POKE μπορεί να χρησιμοποιηθεί εναλλακτικά.
	αλλαγή της διαδρομής (διαρκεί 1bytes)	ΟΧΙ , επειδή το SETUPSP κάνει περισσότερα πράγματα στους εσωτερικούς πίνακες για να κάνει τη διαδρομή να λειτουργήσει.

Παράδειγμα:

Σε αυτό το παράδειγμα δώσαμε στο sprite 31 την εικόνα ενός πλοίου που είναι συγκεντρωμένο στη διεύθυνση &a2f8.

ship = &a2f8

|SETUPSP, 31, 9, κλίτος

Υπάρχει ένας ευκολότερος τρόπος για να καθορίσετε την εικόνα για το sprite χρησιμοποιώντας την IMAGE_LIST στο αρχείο images_yourgame.asm. Εάν έχουμε το NAVE στο IMAGE_LIST, μπορούμε να συσχετίσουμε ένα αναγνωριστικό μεταξύ 16 και 255

|SETUPSP,31, 9, 16 : rem το 16 είναι το αναγνωριστικό του ΠΛΟΙΟΥ στην IMAGE LIST

IMAGE_LIST

;

Θα βάλουμε εδώ μια λίστα με τις εικόνες που θέλουμε για χρησιμοποίησουμε χωρίς να καθορίσουμε τη διεύθυνση μνήμης από τη βασική.

Η εντολή |SETUPSP,<id>,9,<διεύθυνση> γίνεται |SETUPSP,<id>,9,<αριθμός> έτσι η εντολή |SETUPSP,<id>,9,<διεύθυνση> γίνεται |SETUPSP,<id>,9,<αριθμός>.

Το πλεονέκτημα της μη χρήσης διευθύνσεων μνήμης στη BASIC είναι ότι αν μεγεθύνουμε τα γραφικά ή τα επανασυνθέσουμε σε

Ο αριθμός που αναθέτουμε δεν θα αλλάξει.

ΔΕΝ χρειάζεται να έχουν όλα έναν αριθμό, μόνο αυτά που θα χρησιμοποιήσουμε με |setupsp, id, 9,<num>.

Η αριθμητη αρχίζει από το 16

Μπορούμε για χρησιμοποίησουμε έως και 255 εικόνες που καθορίζονται με αυτόν τον τρόπο.

Ο κατάλογος δεν χρειάζεται να έχει μήκος 255 στοιχείων, το μήκος του μπορεί να είναι μεταβλητό, μπορεί ακόμη και να είναι άδειος.

;

DW NAVE ; 16

DW OTHER_SHIP ; 17

;----- BEGIN IMAGE -----

284

NAVE

db 7 ; πλάτος

db 12 ; ύψος

db 0 , 0 , 0 , 0 , 0 , 0 , 0
db 0 , 0 , 0 , 0 , 0 , 0 , 0
db 0 , , 48 , 0 , 0 , 0 , 0
db 0 , , 240 , 48 , 0 , 0 , 0
db 0 , , 207 , 112 , 12 , 0 , 0
db 0 , 84 , 240 , 48 , 164 , 8 , 0
db 0 , 0 , 48 , 176 , 112 , 12 , 0
db 0 , 69 , 48 , 112 , 48 , 101 , 0
db 0 , 16 , 48 , 207 , 207 , 0 , 0
db 0 , , , 80 , 0 , 0 , 0
db 0 , 0 , 0 , 0 , 0 , 0 , 0
db 0 , 0 , 0 , 0 , 0 , 0 , 0

Στην περίπτωση param_number=5, μπορούμε να συμπεριλάβουμε το Vx ως παράμετρο στο τέλος:

|SETUPSP, 31, 5, Vy, Vx

Με αυτόν τον τρόπο θα ενημερώσουμε και τις δύο ταχύτητες με μία εντολή, η οποία κοστίζει

3,73ms σε αντίθεση με τα 6,8ms που θα χρειαζόταν για την κλήση δύο εντολών ξεχωριστά.

Στην περίπτωση χρήσης του param_number=7, εκτός από την αλλαγή της ακολουθίας κινούμενων σχεδίων, η εντολή ενημερώνει αυτόματα το πλαίσιο (frame id), τοποθετώντας το στο αρχικό (το μηδέν) και η διεύθυνση της εικόνας ενημερώνεται, οπότε δεν χρειάζεται να επικαλεστείτε το param_number=9 για να αλλάξετε την εικόνα του sprite στην πρώτη εικόνα της νέας ακολουθίας που έχει οριστεί. Εάν χρησιμοποιείτε |ANIMALL πριν από την εκτύπωση ή

|Ακόμη κι αν το SETUPSP τοποθετήσει το animation στο καρέ μηδέν, θα μεταβείτε στο καρέ 1 πριν την εκτύπωση. Αυτό κανονικά δεν αποτελεί πρόβλημα, αλλά στην περίπτωση μιας ακολουθίας θανάτου, όπου για παράδειγμα το πρώτο καρέ είναι για να διαγράψετε το sprite, μπορεί να μην θέλετε να μεταπηδήσετε απευθείας στο καρέ 1. Σε αυτή την περίπτωση ένα απλό τέχνασμα είναι να επαναλάβετε το καρέ μηδέν στον ορισμό της ακολουθίας θανάτου. Με αυτόν τον τρόπο εξασφαλίζετε ότι το καρέ είναι ορατό. Μια άλλη επιλογή είναι να αφαιρέσετε τη σημαία κίνησης και να το κινήσετε με το ANIMASP μετά την εκτύπωση.

Στην περίπτωση του param_number=15, εκτός από την ανάθεση της διαδρομής στο sprite στον πίνακα χαρακτηριστικών, η εντολή εκτελεί μια εσωτερική επαναφορά δεδομένων, ώστε το sprite να αρχίσει να διατρέχει τη διαδρομή αυτή από το πρώτο τμήμα της εν λόγω διαδρομής.

21.1.25 |STARS

Μετακινεί μια σειρά από έως και 40 αστέρια στην οθόνη (εντός των ορίων που ορίζει η |SETLIMITS), χωρίς να ζωγραφίσει πάνω σε άλλα sprites που έχουν ήδη εκτυπωθεί.

|STARS,<αρχικό αστεριών>,<χρώμα>,<dy>,<dx>,<dx>. **αστέρι>,<αριθμός αστεριών>**

Παράδειγμα:

|STARS, 0, 15, 3, 1, 0

Το παρόδειγμα μετατοπίζει 15 αστέρια χρώματος 3 (κόκκινο) κατά ένα εικονοστοιχείο κάθετα (αφού dy=1 και dx=0). Η επαναλαμβανόμενη κλήση δίνει την αίσθηση ενός κυλιόμενου φόντου αστεριών. Όταν ένα αστέρι φεύγει από το όριο της οθόνης ή από το όριο που έχει οριστεί από το |SETLIMITS, επανεμφανίζεται στην αντίθετη πλευρά, έτσι ώστε να υπάρχει μια αίσθηση συνέχειας στη ροή των αστεριών.

Η τράπεζα αστέρων βρίσκεται στη διεύθυνση 42540 (=&A62C) και έχει χωρητικότητα 40 αστέρων μέχρι τη διεύθυνση 42619. Κάθε αστέρι καταναλώνει 2 bytes, ένα για τη συντεταγμένη Y και ένα για τη συντεταγμένη X.

Οι ομάδες αστεριών μπορούν να μετακινηθούν ξεχωριστά, ξεκινώντας από το αστέρι

της επιλογής σας. Οι αρχικές συντεταγμένες των αστέρων πρέπει να αρχικοποιηθούν από τον προγραμματιστή.

Παράδειγμα αρχικοποίησης και χρήσης σε μια κύλιση τεσσάρων αστρικών επιπέδων για να δοθεί η αίσθηση του βάθους. Κάθε αεροπλάνο θα κινείται με διαφορετική ταχύτητα

1 MNHMH 24999

10 CALL & 6b78: rem εγκαθιστά εντολές RSX

20 bank=42540

30 FOR star=0 TO 39: ' βρόχος για τη δημιουργία 40 αστέρων

40 POKE πάγκος+αστέρι*2,RND*200

50 POKE πάγκος+αστέρι*2+1,RND*80

60 ΕΠΟΜΕΝΟ

70 ΛΕΙΤΟΥΡΓΙΑ 0

80 REM Θα ζωγραφίσουμε και θα μετακινήσουμε 4 αεροπλάνα με 10 αστέρια το καθένα.

90 |STARS,0,10,3,0,-1: ' Το 3 είναι κόκκινο. Τα πιο απομακρυσμένα κινούνται περισσότερο

αργά

91 |STARS,10,10,2,0,-2: ' Το 2 είναι μπλε

92 |STARS,20,10,1,0,-3: ' Το 1 είναι κίτρινο

93 |STARS,30,10,4,0,-4: ' το 4 είναι λευκό. Τα πιο κοντινά είναι πιο γρήγορα

95 goto 90

Οι χρήσεις αυτής της εντολής μπορεί να είναι πολύ διαφορετικές.

- Η ταυτόχρονη χρήση πολλών αστέρων με διαφορετική ταχύτητα και χρώμα μπορεί να δώσει την αίσθηση του βάθους.
- Αν η κατεύθυνση των αστεριών είναι διαγώνια, μπορείτε να δημιουργήσετε ένα "εφέ βροχής".
- Εάν το χρώμα είναι μαύρο και το φόντο είναι καφέ ή πορτοκαλί, μπορείτε να δώσετε την εντύπωση ότι προχωράτε σε αιμώδες έδαφος.
- Αν η κίνηση είναι κυλιόμενη και το χρώμα των αστεριών είναι λευκό, μπορείτε να δώσετε την εντύπωση χιονιού. Η κυλιόμενη κίνηση μπορεί να επιτευχθεί με ένα ζιγκ-ζαγκ στο X διατηρώντας την ταχύτητα στο Y, ή ακόμη και με τη χρήση τριγωνομετρικών συναρτήσεων ό π ως το συνημίτονο. Προφανώς αν χρησιμοποιήσετε το συνημίτονο στη λογική του παιχνιδιού θα είναι πολύ αργό, αλλά μπορείτε να αποθηκεύσετε την προ-υπολογισμένη τιμή του συνημιτόνου σε έναν πίνακα. Παράδειγμα εφέ χιονιού:

1 ΜΝΗΜΗ 23999: ΛΕΙΤΟΥΡΓΙΑ 0

30 ' αρχικοποίηση της τράπεζας 40 αστέρων

40 FOR dir=42540 TO 42619 BHMA 2

45 POKE dir,RND*200:POKE dir+1,RND*80

48 ΕΠΟΜΕΝΟ

50 |STARS,0,20,4,2,dx1

60 |STARS,20,20,4,1,dx2

61 dx1=1*COS(i):dx2=SIN(i)

69 i=i+1: EAN i=359 ΤΟΤΕ i=0

70 GOTO 50

Υπάρχει ένας τρόπος για να επιτύχετε ταχύτερη εκτέλεση, και αυτός είναι να αποφύγετε το πέρασμα παραμέτρων. Σε όλο αυτό το βιβλίο είδαμε πως το πέρασμα παραμέτρων είναι ακριβό ακόμα και αν η εντολή που καλείται δεν κάνει τίποτα. Λοιπόν, έχουμε να κάνουμε με μια εντολή που απαιτεί 5 παραμέτρους, οπότε είναι ιδιαίτερα ακριβή. Αν θέλουμε να μειώσουμε το χρόνο που χρειάζεται η BASIC για να ερμηνεύσει τις παραμέτρους, μπορούμε απλά να καλέσουμε την εντολή μία φορά με παραμέτρους και τις επόμενες φορές χωρίς να περάσουμε παραμέτρους.

| STARS,0,10,1,1,5,0

| STARS : αυτή η κλήση χωρίς παραμέτρους λαμβάνει τις ίδιες τιμές με την τελευταία κλήση.

Αυτή η δυνατότητα είναι ιδιαίτερα χρήσιμη σε παιχνίδια όπου θέλουμε να επικαλούμαστε την εντολή σε κάθε κύκλο παιχνιδιού για να μετακινήσουμε τα αστέρια, καθώς θα εξοικονομήσει περίπου 1,7 ms.

ΣΗΜΑΝΤΙΚΟ: Η εντολή STARS επηρεάζεται από τα όρια |SETLIMITS, αλλά μόνο εάν τα Vx ή Vy είναι μη μηδενικά. Αν και τα δύο είναι μηδέν, τότε η εντολή |SETLIMITS δεν επηρεάζεται και τα αστέρια μπορούν να ζωγραφιστούν σε όλη την οθόνη.

21.1.26 |UMAP

Αυτή η εντολή ενημερώνει τον χάρτη με πληροφορίες που βρίσκονται σε μια άλλη περιοχή μνήμης όπου έχουμε έναν μεγαλύτερο χάρτη. Η εντολή προκαλεί την αναδημιουργία ολόκληρου του χάρτη, περιλαμβάνοντας μόνο τα στοιχεία που ανταποκρίνονται σε συγκεκριμένες περιοχές συντεταγμένων X, Y (όλες οι παράμετροι είναι αριθμοί 16-bit).

Χρήση:

|UMAP, <map_ini>, <map_fin>, <y_ini>, <y_fin>, <x_ini>, <x_fin>, <x_fin>, <x_fin>, <x_fin>, <x_fin>, <x_fin>, <x_fin>, <x_fin>, <x_fin>, <x_fin>.

Για παράδειγμα, αν έχουμε έναν χάρτη που βρίσκεται στη διεύθυνση 22.000 και καταλαμβάνει 1500bytes και θέλουμε να ενημερώσουμε τον χάρτη με τις συντεταγμένες του χαρακτήρα μας, με αρκετό περιθώριο για να προχωρήσουμε στη συντεταγμένη Y μέχρι 100 γραμμές και στη συντεταγμένη x μέχρι 20 bytes προς όλες τις κατευθύνσεις:

|UMAP, 22000, 23500, y-100, y+100, x-20, x+20

Αυτή η εντολή θα ελέγξει τις συντεταγμένες των στοιχείων που βρίσκονται στο χάρτη στη διεύθυνση 23000 και αν βρίσκονται εντός των περιθωρίων X, Y που έχουμε ορίσει, θα αντιγραφούν στην περιοχή μνήμης που χρησιμοποιεί η 8BP για την εντολή |MAP2SP, δηλαδή θα τα αντιγράψει από τη διεύθυνση 42040. Ωστόσο, θα αντιγράψει μόνο αυτά που πληρούν τη συνθήκη. Καθώς υπάρχουν λιγότερα στοιχεία, η εντολή |MAP2SP θα εκτελεστεί ταχύτερα, καθώς θα πρέπει να διαβάσει και να ελέγξει αν υπάρχουν λιγότερα στοιχεία στην οθόνη.

ΣΗΜΑΝΤΙΚΟ: ο προς αντιγραφή χάρτης ΔΕΝ πρέπει να περιλαμβάνει τις 3 παραμέτρους κάθε χάρτη:

<Max_high> (το οποίο είναι DW , δηλαδή 2 bytes)
<max_width> (το οποίο είναι ένα DW , δηλαδή 2 bytes)
<Num_items> (που είναι μια DB , δηλαδή 1 byte)

Δηλαδή, υπάρχουν 5 bytes που δεν πρέπει να περιλαμβάνονται στο χάρτη που πρόκειται να αντιγραφεί.

22 Πώς να φτιάξετε έναν πίνακα αποτελεσμάτων

Σε πολλά παιχνίδια είναι ενδιαφέρον να υπάρχει ένας μηχανισμός πίνακα αποτελεσμάτων (που ονομάζεται επίσης "Hall of fame"), ο οποίος αποθηκεύει τα καλύτερα σκορ από διάφορα παιχνίδια με τακτικό τρόπο. Μπορεί να προγραμματιστεί στη BASIC μέσω ενός πίνακα που αποθηκεύει τη βαθμολογία κάθε παιχνιδιού, αλλά κάθε φορά που σταματάμε το παιχνίδι και RUN, ο διερμηνέας BASIC εκτελεί εσωτερικά ένα CLEAR και όλες οι τιμές αυτού του πίνακα διαγράφονται. Ένας τρόπος για να το αποφύγουμε αυτό είναι να εμποδίσουμε τον χρήστη να διακόψει το πρόγραμμα πατώντας δύο φορές το πλήκτρο ESC με τη χρήση της ρουτίνας του firmware CALL &bb48. Μια άλλη επιλογή είναι να αποθηκεύσετε τα σημεία σε έναν πίνακα στη μνήμη και να τον διαβάσετε και να τον τροποποιήσετε από τη BASIC. Αυτό μπορεί να προγραμματιστεί με πολλούς τρόπους, και εδώ είναι ένα παράδειγμα. Τα βήματα που πρέπει να γίνουν είναι τα εξής:

Συμπεριλάβετε στο make_all.asm την ανάγνωση του αρχείου "score_table.asm".

```
;----- CODIGO -----  
;περιλαμβάνει τη βιβλιοθήκη 8bp και το music  
playerWYZ διαβάστε το  
"make_codigo_mygame.asm".  
;----- MUSICA -----  
διαβάστε το  
"make_musica_mygame.asm",  
περιλαμβάνει τα τραγούδια.  
; ----- ΓΡΑΦΗΜΑΤΑ -----  
;αυτό το μέρος περιλαμβάνει εικόνες και ακολουθίες  
κινουμένων σχεδίων, διαβάστε το  
"make_graficos_mygame.asm".  
διάβασε "score_table.asm"
```

Στη συνέχεια, πρέπει να δημιουργήσουμε ένα τέτοιο αρχείο score_table.asm με δειγματικά δεδομένα. Δημιούργησα ένα με ονόματα Σουμερίων θεών και βαθμολογίες από 10 έως 1. Κάθε όνομα χρειάζεται 8 χαρακτήρες. Κάτι πολύ σημαντικό είναι το όργανο _end_graph. Με την εντολή αυτή δηλώνουμε ότι ο πίνακας θα συναρμολογηθεί μετά τα γραφικά, στις διευθύνσεις μνήμης που ακολουθούν.

```
org _end_graph  
_SCORE_TABLE  
db "ISHTAR "  
dw 10 db  
"ANTU "  
dw 9  
db "INANNA "  
dw 8  
db "NIMUG "  
dw 7  
db "NIMBARA "  
dw 6  
db "ASTA "  
dw 5  
db "DAMKINA "  
dw 4  
db "NEBAT "  
dw 3  
db "NISABA "  
dw 2  
db "NINSUB "  
dw 1  
_END_SCORE_TABLE
```

Τώρα έρχεται το BASIC μέρος: Θα συναρμολογήσουμε με το winape και στη συνέχεια θα δούμε (με το winape, επιλογή assemble->symbols) σε ποια διεύθυνση μνήμης έχει συναρμολογηθεί η ετικέτα end_graph. Στην περίπτωσή μου ήταν η &9685. Στο πρόγραμμα BASIC θα θα λάβουμε υπόψη (το αποθηκεύω στη μεταβλητή "dir")

```

190 ' --- αίθουσα φήμης
200 DIM pts(11): DIM name$(11):'σκορ
210 GOSUB 2040: "ανάγνωση πίνακα αποτελεσμάτων
220 INK 3,7: PEN 3: LOCATE 15,12: PRINT " Hall of fame ": LOCATE 15,13: PRINT
" -----
230 p=1:FOR i=0 TO 9:LOCATE 1,i+14: PEN p :PRINT , name$(i), pts(i) :
p=1+(p MOD 3):NEXT
240 b$=INKEY$:IF b$="" THEN 240 ELSE 250

```

Ας δούμε τη ρουτίνα που διαβάζει τον πίνακα αποτελεσμάτων στη γραμμή 2040.

```

2040 '--- ΔΙΑΒΑΣΕ ΤΟΝ ΠΙΝΑΚΑ
    ΑΠΟΤΕΛΕΣΜΑΤΩΝ
2050 dir=&9685: FOR i=0 TO 9: name$(i)=""
2070 FOR j=dir TO dir +7: 'Ιες χαρακτήρας ένας      τα 8 γράμματα
    χαρακτήρας
2080 letter=PEEK (j):
    name$(i)=name$(i)+CHR$(letter)
2090 NEXT j: dir=dir+8: 'μετά τα 8 γράμματα      σημεία (ακέραιος
    υπάρχουν τα αριθμός)
2100 pts(i)=0: | PEEK, dir, @pts(i): dir=dir+2: 'ένας ακέραιος αριθμός είναι 2
    bytes
2110 ΕΠΟΜΕΝΟ i
2120 RETURN

```

Τέλος, κάθε φορά που τελειώνει ένα παιχνίδι, ελέγχουμε αν το σκορ (μεταβλητή "score" στο παράδειγμα) είναι υψηλότερο από οποιαδήποτε εγγραφή στον πίνακα σκορ (πίνακας "pts") και αν ναι, τροποποιούμε τον πίνακα. Τροποποιώντας τον πίνακα στις διευθύνσεις μνήμης, δεν θα χάσουμε τις τιμές, ακόμα και αν RUN.

```

1800 '--- ΤΕΛΟΣ ΠΑΙΧΝΙΔΙΟΥ & ΕΛΕΓΧΟΣ ΥΨΗΛΟΥ ΣΚΟΡ ---
1810 INK 0,0:BORDER 5: INK 2,15:INK 1,20: | MUSIC
1820 j=10:FOR i=9 TO 0 STEP -1: IF score>pts(i) THEN j=i:NEXT
1830 ΑΝ j=10 ΤΟΤΕ RUN: 'τέλος παιχνιδιού & έναρξη
1831 μετακινεί όλα τα χαμηλότερα σκορ κατά μία θέση.
1840 FOR i=8 TO j STEP -1: pts(i+1)=pts(i): name$(i+1)=name$(i): NEXT
1850 b$=INKEY$:IF b$<>"" THEN 1850: 'καθαρισμός του πληκτρολογίου buffer
1860 MODE 1: BORDER 5: INK 3,8: LOCATE 6,8: PEN 3: PRINT "CONGRATULATIONS!
ΝΕΟ ΥΨΗΛΟ ΣΚΟΡ".
1880 ΕΝΤΟΠΙΣΤΕ      2:PRINT "ENTER YOUR NAME".
    14,10: PEN
1900 ΕΝΤΟΠΙΣΤΕ      1:INPUT name$(j): name$(j)=MID$(name$(j),1,8)
    15,12: PEN
1910 pts(j)=score
    '--- ΓΡΑΨΤΕ      ΠΙΝΑΚΑΣ ΓΙΑ ΤΗ ΜΝΗΜΗ --
    ΣΚΟΡ
1930 dir=&9685: FOR i=0 TO 9: k=1
1950 FOR j=dir TO dir +7: 'γράφουμε χαρακτήρα προς χαρακτήρα,
    και τα 8 γράμματα
1960 dato$=MID$(name$(i),k,1): IF dato$="" THEN dato$=" "
1970 dato=ASC(dato$)
    POKE j,data:k=k+1:NEXT j

```

1990 dir=dir+8: 'γράφουμε τα σημεία στίξης μετά το όνομα (8 γράμματα)
2000 |POKE,dir,pts(i)
2010 dir=dir+2:'η βαθμολογία είναι ακέραιος αριθμός = 2 bytes
2020 ΕΠΟΜΕΝΟ i
2030 RUN

23 Πιθανές μελλοντικές βελτιώσεις της βιβλιοθήκης

Η βιβλιοθήκη 8BP μπορεί να βελτιωθεί με την προσθήκη νέων λειτουργιών που θα μπορούσαν να ανοίξουν νέες δυνατότητες για τον προγραμματιστή. Ακολουθούν ορισμένες προτάσεις για το σκοπό αυτό

23.1 Μνήμη για τον εντοπισμό νέων λειτουργιών

Επί του παρόντος, μέσω του μηχανισμού των "επιλογών συναρμολόγησης", η βιβλιοθήκη σας αφήνει ένα ποσό ελεύθερης μνήμης για την καταχώριση BASIC που εξαρτάται από κάθε επιλογή.

- Επιλογή 0: 23,5 KB δωρεάν (θα πρέπει να χρησιμοποιείται μόνο για δοκιμές)
- Επιλογή 1: 25 KB δωρεάν (παιχνίδια λαβύρινθου)
- Επιλογή 2: 24,8 KB δωρεάν (παιχνίδια με κύλιση)
- Επιλογή 3: 24 KB δωρεάν (παιχνίδια με ψευδο 3D)

Η βιβλιοθήκη θα μπορούσε ακόμη να αναπτυχθεί, μέσω μιας επιλογής 4 που επεκτείνει τις δυνατότητες της επιλογής 1, π.χ. μέσω δυνατοτήτων "κινηματογράφησης". Αυτή η επιλογή 4 θα μπορούσε να παρέχει τέτοιες δυνατότητες χρησιμοποιώντας 1 KB και αφήνοντας στον χρήστη 24 KB ελεύθερα.

23.2 Εκτύπωση ανάλυσης pixel

Επί του παρόντος, η 8BP χρησιμοποιεί ανάλυση byte και συντεταγμένες byte, οι οποίες είναι 2 pixel της λειτουργίας 0. Στην πραγματικότητα, ένας τρόπος για να παρακάμψετε αυτόν τον περιορισμό είναι να ορίσετε 2 εικόνες για το ίδιο sprite που είναι μετατοπισμένες κατά ένα μόνο pixel. Κατά τη μετακίνηση του sprite στην οθόνη μπορείτε να εναλλάσσετε μεταξύ της απλής αλλαγής της εικόνας στην εικόνα με τη μετατόπιση και της μετακίνησης του sprite κατά ένα byte. Με αυτόν τον τρόπο θα έχετε μια κίνηση pixel-by-pixel. Αυτή η τεχνική περιγράφεται λεπτομερώς στο κεφάλαιο 13

23.3 Διάταξη λειτουργίας 1

Η τρέχουσα διάταξη λειτουργεί ως απομονωτής χαρακτήρων $20 \times 25 = 500$ Bytes.

Μπορεί να χρησιμοποιηθεί σε παιχνίδια με λειτουργία 1 χωρίς προβλήματα, αλλά θα υπάρχουν πράγματα που δεν μπορούμε να κάνουμε, όπως ο ορισμός ενός μέρους που καταλαμβάνει 3 χαρακτήρες πλάτους λειτουργίας 1, καθώς οι χαρακτήρες λειτουργίας 0 καταλαμβάνουν το διπλάσιο πλάτος από τους χαρακτήρες λειτουργίας 1. Δεν είναι πρόβλημα, αλλά είναι ένας περιορισμός.

Μια διάταξη λειτουργίας 1 θα καταλαμβάνει 1KB, οπότε $40 \times 25 = 1000$. Δεδομένου ότι η διάταξη mode 0 και η διάταξη mode 1 δεν θα χρησιμοποιούνται ταυτόχρονα, θα μπορούσαν να επικαλύπτονται στη μνήμη και λαμβάνοντας υπόψη ότι η διάταξη mode 0 βρίσκεται στο 42000 έως 42500, θα τοποθετούσαμε απλώς τη διάταξη mode 1 μεταξύ 41500 και 42500, "κλέβοντας" 500 bytes από τη μνήμη sprite των 8KB, που βρίσκεται μεταξύ 34000 και 42000.

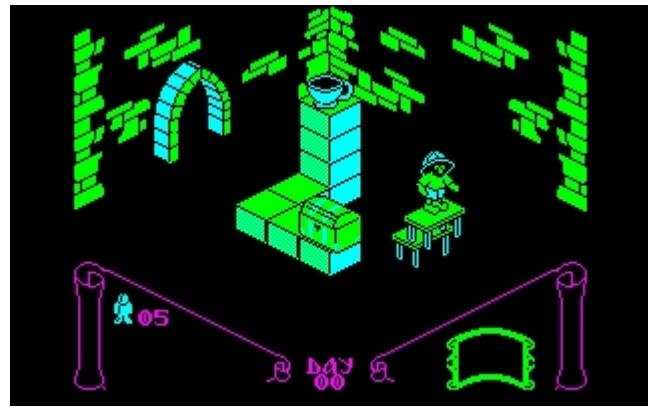
Οι αλλαγές για την υποστήριξη αυτής της βελτίωσης είναι ελάχιστες, επηρεάζοντας μόνο δύο χαρακτηριστικά

Οι διατάξεις layout0/layout1 και | LAYOUT και | COLAY θα πρέπει να γνωρίζουν τον τρόπο λειτουργίας της οθόνης, μέσω μιας μεταβλητής που λειτουργεί ως σημαία (layout0/layout1). Αυτή η τροποποίηση θα μπορούσε να είναι μια χαρά, αλλά ακόμα και χωρίς αυτήν, μπορούμε να χρησιμοποιούμε διατάξεις σε παιχνίδια με τρόπο

λειτουργίας 1 χωρίς προβλήματα.

23.4 Ικανότητα κινηματογράφησης

Θα ήταν ενδιαφέρον να δημιουργηθεί μια λειτουργία "filmation" για την κατασκευή παιχνιδιών τύπου "knight lore". Κάνοντας χρήση των υπαρχουσών συναρτήσεων της βιβλιοθήκης, με λίγο επιπλέον κώδικα θα μπορούσε να υλοποιηθεί αυτή η ενδιαφέρουσα δυνατότητα. Είναι πιθανόν αυτή η δυνατότητα να προστεθεί σύντομα.



Εικ. 120 το μυθικό "Knight Lore".

23.5 Λειτουργίες κύλισης υλικού

Υπάρχουν λίγα παιχνίδια Amstrad CPC με ποιοτική ομαλή κύλιση, προγραμματισμένα με τις δυνατότητες του τσιπ ελέγχου βίντεο M6845. Επί του παρόντος, η βιβλιοθήκη διαθέτει έναν μηχανισμό κύλισης βασισμένο στη CPU (όχι βασισμένο στο υλικό, αλλά αποτελεσματικό και ευέλικτο για παιχνίδια με κύλιση προς οποιαδήποτε κατεύθυνση κίνησης).

Το γεγονός ότι δεν υπάρχουν πολλά τέτοια παιχνίδια οφείλεται στο ότι τη δεκαετία του 1980 οι προγραμματιστές βιντεοπαιχνιδών δεν είχαν πολλές πληροφορίες και σε πολλές περιπτώσεις ήταν ερασιτέχνες.

Μεταξύ των λίγων παιχνιδιών που έχουν ομαλή κύλιση είναι 2 από την Firebird:

- "Αποστολή Γενοκτονία" (από το Firebird, 1987, από τον Paul Shirley, έναν εξαιρετικό προγραμματιστή που εφηύρε επίσης μια εξαιρετικά γρήγορη τεχνική αντικατάστασης χωρίς τη χρήση μασκών).
- "Warhawk" (από Firebird, 1987)



Εικ. 121 Παιχνίδια Firebird με γρήγορη κύλιση

Η τεχνική κύλισης αυτών των δύο παιχνιδιών είναι η ίδια, γνωστή ως "κάθετη κύλιση". Η τεχνική κύλισης συνίσταται στον ακριβή έλεγχο της στιγμής κατά την οποία πραγματοποιείται η κύλιση της οθόνης. Εκείνη τη στιγμή ξεγελάμε το CTRC 6845 λέγοντάς του ότι η οθόνη τελειώνει νωρίτερα από το κανονικό. Ωστόσο, πριν τελειώσει αυτό το τμήμα της οθόνης, του λέμε να ενσωματώσει λιγότερες γραμμές σάρωσης από όσες αντιστοιχούν σε ένα τμήμα αυτού του μεγέθους. Στη συνέχεια, σε μια πολύ ακριβή στιγμή που πρέπει να ελέγξουμε με ακρίβεια μικροδευτερολέπτου, λέμε στο τσιπ να

ξεκινήσει μια νέα οθόνη, χωρίς να έχει παραχθεί το σήμα κατακόρυφου συγχρονισμού.
Αυτό μας επιτρέπει να σχεδιάσουμε μια δεύτερη

οθόνη (για παράδειγμα, οι δείκτες) και αντισταθμίστε τον αριθμό των γραμμών σάρωσης του πρώτου τμήματος. Αν πετύχουμε σωστά τον μηχανισμό αντιστάθμισης γραμμών σάρωσης, μπορούμε να κάνουμε το ένα από τα δύο τμήματα της οθόνης να κινηθεί εξαιρετικά ομαλά. Το πρόβλημα με την μεταφορά αυτής της τεχνικής σε μια εντολή που θα χρησιμοποιηθεί από τη BASIC είναι ότι ο έλεγχος των διακοπών είναι ανακριβής λόγω της εκτέλεσης του διερμηνέα, και εδώ χρειαζόμαστε πολύ, πολύ ακριβή έλεγχο.

Το πρόβλημα με την κύλιση υλικού (η οποία επηρεάζει επίσης την κύλιση λογισμικού που μετακινεί ολόκληρη την οθόνη) είναι ότι "σέρνει" τα sprites που υπάρχουν μαζί του, οπότε όταν τα επανατοποθετείτε θα παρατηρήσετε μια ανεπιθύμητη δόνηση στους εχθρούς ή/και στον χαρακτήρα σας. Για να το λύσετε αυτό, μπορείτε να χρησιμοποιήσετε διπλό buffering και να εναλλάσσεστε μεταξύ δύο μπλοκ των 16KB κάθε φορά που είναι έτοιμο ένα καρέ. Αυτό θα σας αποτρέψει από το να μπορείτε να βλέπετε "πώς γίνεται κάθε καρέ". Στο 8BP απέρριψα το double buffering προκειμένου να αφήσω έναν καλό χώρο RAM για τον προγραμματιστή και γι' αυτό δεν έχουν εφαρμοστεί αυτές οι τεχνικές.

Για όλους τους παραπάνω λόγους, η κύλιση στο 8BP βασίζεται σε έναν παγκόσμιο χάρτη που δεν σέρνει τα sprites κατά την κίνηση και επομένως είναι πιο αποδοτική, καθώς μετακινεί λιγότερη μνήμη, ενώ επιτρέπει την πολυκατευθυντική κίνηση.

23.6 Μεταφορά της βιβλιοθήκης 8BP σε άλλους μικρούπολογιστές

Αυτή η βιβλιοθήκη θα μπορούσε να μεταφερθεί εύκολα σε άλλους μικρούπολογιστές με βάση τον Z80, όπως ο Sinclair ZX Spectrum. Στην περίπτωση του ZX Spectrum θα ήταν απαραίτητο να ξαναγράψετε τις ρουτίνες που ζωγραφίζουν στην οθόνη, καθώς η μνήμη βίντεο χειρίζεται διαφορετικά. Η μετάβαση στο ZX είναι ήδη ένα σταθερό σχέδιο, αφού έχει δεχτεί πολλά αιτήματα από χρήστες του ZX.

Η μεταφορά της βιβλιοθήκης σε έναν Commodore 64 θα ήταν επίσης εφικτή, αν και ο κώδικας συναρμολόγησης δεν θα μπορούσε να επαναχρησιμοποιηθεί, δεδομένου ότι βασίζεται σε άλλο μικροεπεξεργαστή. Επιπλέον, στην περίπτωση του Commodore 64, η μετάβαση της βιβλιοθήκης 8BP θα πρέπει να εκμεταλλευτεί τα ίδια τα χαρακτηριστικά της μηχανής, όπως τα 8 sprites υλικού, έτσι ώστε αυτό που θα πρέπει να ενσωματώσει εσωτερικά η βιβλιοθήκη 8BP να είναι ένας πολυπλέκτης sprite, προσφέροντας 32 sprites, αλλά χρησιμοποιώντας εσωτερικά τα 8 sprites υλικού.



Εικ. 122 Sinclair ZX και Commodore 64, δύο κλασικά μοντέλα

24 Μερικά παιχνίδια που έγιναν με την 8BP

Σε αυτό το κεφάλαιο θα περιγράψω τον τρόπο με τον οποίο κατασκευάζονται μερικά παιχνίδια που μπορείτε να βρείτε στον ιστότοπο <https://github.com/jjaranda13/8BP> που έχουν φτιαχτεί με το 8BP (από τα πιο πρόσφατα έως τα παλαιότερα):

- **Paco the man:** ένα παιχνίδι τύπου pacz, το οποίο χρησιμοποιεί την τεχνική soft move (μισό byte) και προηγμένη μαζική λογική.
- **NOMWARS:** ένα παιχνίδι τύπου "commando"
- **Blaster pilot:** ένα παιχνίδι κύλισης πολλαπλών κατευθύνσεων και είναι εμπνευσμένο από το στυλ παιχνιδιών όπως το "Time Pilot" ή το "Asteroids".
- **Happy Monty:** γρήγορο παιχνίδι σε στυλ μεταλλαγμένου Monty
- **Eridu:** ένα κλασικό παιχνίδι τύπου scramble με οριζόντια κύλιση.
- **Διαστημικό φάντασμα:** εμπνευσμένο από το space harrier
- **Eternal Frogger:** ένα remake του κλασικού frogger, που παρουσιάστηκε στην περίφημη έκθεση "eternal amstrad".
- **3D Racing one:** το πρώτο παιχνίδι αγώνων που χρησιμοποιεί ψευδο-τρισδιάστατη ικανότητα
- **Φρέσκα φρούτα και λαχανικά:** ένα παιχνίδι πλατφόρμας με οριζόντια κύλιση και προηγμένη διαχείριση διαδρομής sprite.
- **Nibiru:** ένα παιχνίδι οριζόντιας κύλισης πλοιών, που χρησιμοποιεί προηγμένα χαρακτηριστικά του 8BP
- **Anunnaki:** a ship game, είδος arcade
- **Mutante Montoya:** ένα παιχνίδι κύλισης οθόνης. Θα μπορούσε να χαρακτηριστεί ως platformer. Ήταν το πρώτο παιχνίδι που έφτιαξα με το 8BP.
- **Μίνι-παιχνίδια:** πρόκειται για σύντομα, απλά, διδακτικά παιχνίδια για να ξεκινήσετε τον προγραμματισμό με το 8BP. Υπάρχει μια έκδοση του κλασικού "pong", που ονομάζεται "Mini-pong" και μια έκδοση του κλασικού "Space Invaders", που ονομάζεται "mini-invaders".

24.1 Μεταλλαγμένος Montoya

Ένας πρώτος φόρος τιμής στο Amstrad CPC, με τίτλο εμπνευσμένο από το κλασικό "mutant monty".

Είναι ένα απλό παιχνίδι 5 επιπέδων. Βασίζεται στη χρήση της διάταξης 8BP για την κατασκευή κάθε οθόνης.





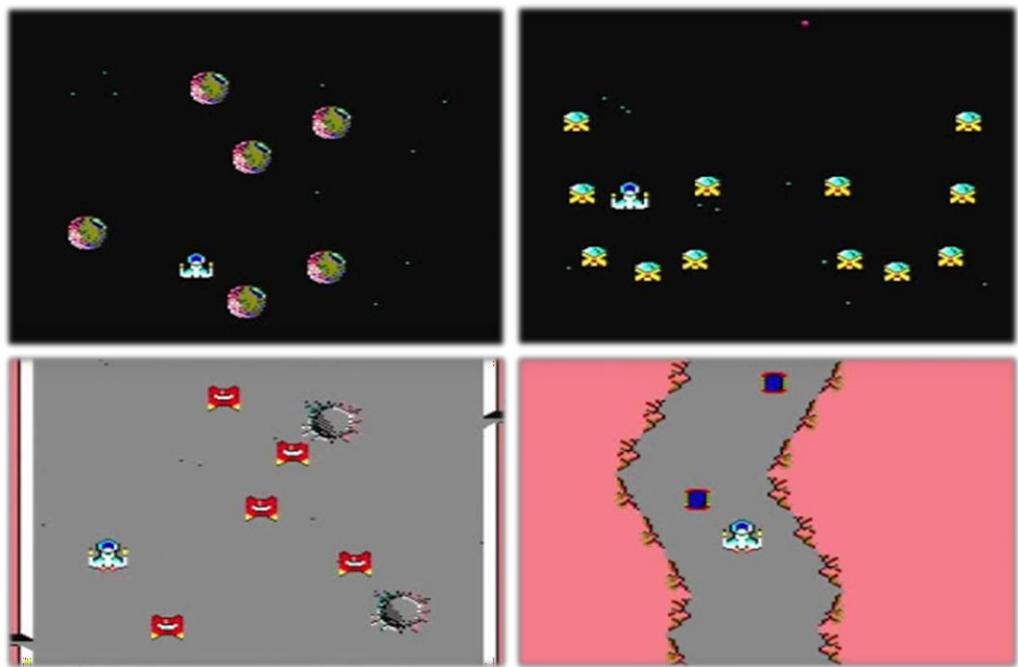
24.2 Anunnaki, το εξωγήινο παρελθόν μας

Πρόκειται για ένα πολύ ενδιαφέρον arcade βιντεοπαιχνίδι για να το αναλύσετε και να μάθετε την τεχνική προγραμματισμού "massive logic". Όταν προγραμματίστηκε, η βιβλιοθήκη 8BP δεν διέθετε ακόμη εντολή κύλισης ή δρομολόγηση sprite, γι' αυτό και ο προγραμματισμός αυτού του παιχνιδιού είναι τόσο ενδιαφέρων, καθώς επιτυγχάνει τα πάντα μέσω της μαζικής λογικής.

Σε αντίθεση με το "Mutant Montoya", το βιντεοπαιχνίδι "Anunnaki" δεν κάνει χρήση της διάταξης, καθώς πρόκειται για ένα παιχνίδι που αφορά την προώθηση και την καταστροφή εχθρικών πλοίων, δεν είναι ένα παιχνίδι λαβύρινθων ή οθονών που περνούν. Αυτό το παιχνίδι κάνει επίσης χρήση του της "προσομοιωμένης" κύλισης, πολύ ενδιαφέρον.

Είστε ο Enki, ένας διοικητής των Anunnaki που αντιμετωπίζει εξωγήινες φυλές για να κατακτήσουν τον πλανήτη Γη και να υποτάξουν έτσι τους ανθρώπους στη θέλησή τους.

Το παιχνίδι αποτελείται από 2 επίπεδα, αν και αν χάσετε μια ζωή, συνεχίζετε στο σημείο του επιπέδου όπου βρίσκεστε, δεν επιστρέφετε στην αρχή του επιπέδου. Το πρώτο επίπεδο είναι ένα στάδιο στο διαστρικό διάστημα, όπου πρέπει να αποφύγετε μετεωρίτες και να σκοτώσετε ορδές πλοίων και διαστημικών πουλιών. Στο τέλος του επιπέδου πρέπει να καταστρέψετε ένα "αφεντικό". Το δεύτερο επίπεδο λαμβάνει χώρα στο φεγγάρι, όπου πρέπει να καταστρέψετε ορδές πλοίων, και στη συνέχεια πρέπει να περάσετε μέσα από ένα τούνελ γεμάτο νάρκες μέχρι να συναντήσετε τρία "αφεντικά" που πρέπει να καταστρέψετε.



24.3 Nibiru

Αυτό είναι ένα παιχνίδι που δοκιμάζει πολλά από τα χαρακτηριστικά του 8BP και της τεχνικής προγραμματισμού "massive logic", και έχει λεπτομέρειες όπως ένα φανταχτερό γράφημα φόρτωσης και τρεις μελωδίες μέσα στο παιχνίδι, καθώς και έναν πίνακα αποτελεσμάτων που δεν χάνεται ακόμα και αν επανεκκινήσετε το παιχνίδι και άλλες προηγμένες τεχνικές πτυχές όπως κύλιση parallax, διαδρομές, μακροεντολές κ.λπ. Η λίστα BASIC είναι μόλις πάνω από 16KB.

Είστε ο πιλότος ενός αντιτορπιλικού σκάφους και πρέπει να νικήσετε τον πλανήτη Nibiru και τον ηγέτη του, τον "Gorgo", ένα σχεδόν ανίκητο αρχαίο ερπετό. Πρέπει να καταστρέψετε τα γαλαξιακά πουλιά που ζουν στα φεγγάρια του και μόλις φτάσετε στον πλανήτη πρέπει να αντιμετωπίσετε τους κινδύνους του πριν μπορέσετε να πολεμήσετε τον Gorgo.

Το παιχνίδι αποτελείται από τρεις φάσεις και χρησιμοποιεί τον μηχανισμό κύλισης της 8BP που βασίζεται στην εντολή MAP2SP και χρησιμοποιεί επίσης δρομολόγηση sprite και μακροεντολές κίνησης - όλα από τη BASIC! χάρη στην 8BP και την τεχνική "massive logic".



Το παιχνίδι διατηρεί έναν πίνακα αποτελεσμάτων που δεν διαγράφεται, ακόμη και αν σταματήσετε το παιχνίδι. Αυτό επιτυγχάνεται αποθηκεύοντάς τον στη μνήμη RAM με

pokes, αντί να τον αποθηκεύετε σε μεταβλητές της BASIC.

24.4 Φρέσκα φρούτα & λαχανικά

Αυτό είναι ένα παιχνίδι πλατφόρμας, όπου η αποστολή σας είναι να συλλέξετε όλα τα φρούτα για να αφήσετε τον πληθυσμό να μην έχει τίποτα να φάει, έτσι ώστε να πρέπει να θυσιάσουν ένα φτωχό γουρούνι για να τραφούν.

Η κύρια καινοτομία του είναι η προηγμένη χρήση των διαδρομών, η σύνδεση της μιας με την άλλη για να περάσουμε από το "πέσιμο" στο "περπάτημα" και η χρήση του RINK σε συνδυασμό με το MAP2SP ως τεχνική κύλισης.



Τα μπλε τούβλα στην παρουσίαση του παιχνιδιού χρησιμοποιούν συγχρονισμένη εκτύπωση sprite (PRINTSPALL,0,0,1), ενώ κατά τη διάρκεια του παιχνιδιού η εκτύπωση sprite δεν είναι συγχρονισμένη (PRINTSPALL,0,0,0,0). Το αποτέλεσμα του συγχρονισμού στην παρουσίαση είναι ότι όλα εμφανίζονται συγχρονισμένα, συμπεριλαμβανομένου του animation RINK ink (που χρησιμοποιείται στα μπλε τούβλα). Αυτό δίνει ένα ομαλό εφέ κύλισης. Κατά τη γνώμη μου, δεν πρέπει να συγχρονίζετε ένα παιχνίδι, διότι, αν και επιτυγχάνεται μεγαλύτερη ομαλότητα, μειώνεται επίσης η ταχύτητα του παιχνιδιού. Ανάλογα με το είδος του παιχνιδιού που παίζετε, μπορεί να σας ενδιαφέρει ή να μην σας ενδιαφέρει.

24.5 "3D Racing one

Αυτό είναι το πρώτο παιχνίδι που δημιουργήθηκε με τη χρήση της δυνατότητας pseudo3D της 8BP. Έχει ένα φανταχτερό γραφικό φόρτωσης και 4 πίστες: μια πίστα εκπαίδευσης με λακκούβες στο δρόμο, μια πίστα όπου ανταγωνίζομαστε με 4 άλλα αυτοκίνητα, μια πίστα όπου η ταχύτητα διπλασιάζεται, χρησιμοποιώντας τμήματα με χαμηλότερη κλίση, και ένα τελικό νυχτερινό στάδιο. Το παιχνίδι χρησιμοποιεί επίσης την εντολή PRINTAT και το δικό του σετ χαρακτήρων. Επιπλέον, διαθέτει πίνακα αποτελεσμάτων, ένα κύριο μουσικό κομμάτι, δύο δευτερεύοντα κομμάτια και ηχητικά εφέ.

Για την παρουσίαση χρησιμοποιείται ένας 2D χάρτης γεμάτος μπάλες και ρυθμίζεται η εντολή MAP2SP με αντικατάσταση. Οι μπάλες είναι "εικόνες ζουμ".

Το πρώτο κύκλωμα έχει κατασκευαστεί με ένα αρχείο στο οποίο έχουμε τοποθετήσει ένα προς ένα όλα τα στοιχεία (τμήματα, πινακίδες, δέντρα, λακκούβες...) αλλά τα

υπόλοιπα κυκλώματα δημιουργούνται δυναμικά από τη BASIC, σκαλίζω τη διεύθυνση του παγκόσμιου χάρτη.

Για την ανίχνευση συγκρούσεων και εξόδου από τον δρόμο, χρησιμοποιείται η εντολή PEEK αντί της COLSPALL, οπότε μόλις ανιχνεύσει ένα byte στο αυτοκίνητο με χρώμα διαφορετικό από το χρώμα του δρόμου, θεωρείται ότι συγκρούεστε και ο κινητήρας σας καταστρέφεται.

Ένα άλλο ενδιαφέρον νέο χαρακτηριστικό είναι η χρήση δυναμικών διαδρομών. Τόσο η διαδρομή του αγωνιστικού κυκλώματος όσο και οι διαδρομές των αυτοκινήτων δημιουργούνται από το BASIC, ακολουθώντας τη διαδικασία που εξηγείται σε αυτό το εγχειρίδιο.



24.6 Διαστημικό φάντασμα

Αυτό είναι ένα παιχνίδι που έγινε με την έκδοση v35 της βιβλιοθήκης. Χρησιμοποιεί δυνατότητες ψευδο-3D για την παρουσίαση τίτλων σε στυλ "Star Wars". Χρησιμοποιεί επίσης διαφανή εκτύπωση με sprites (νομίσματα) που περνούν πίσω από το φόντο (τον πίνακα αποτελεσμάτων).

Το παιχνίδι είναι εμπνευσμένο από το κλασικό "Space Harrier" και ελέγχετε έναν ήρωα εξοπλισμένο με ένα jet-pack που πετάει στο διάστημα, σκοτώνοντας μετεωρίτες, UFO, διαστημικά πουλιά και έναν δράκο ως τελικό εχθρό. Αποτελείται από τρία στάδια και ένα επικό φινάλε.

Το σετ χαρακτήρων είναι εσωτερικό, αν και έχουν οριστεί μόνο οι αριθμοί, στο στυλ ενός "ρολογιού Casio" για τους δείκτες.

Στην πρώτη φάση, χρησιμοποιούνται διαδρομές για τα αστέρια, τα οποία είναι sprites, καθώς και για τους μετεωρίτες και τα πουλιά.

Το δεύτερο χρησιμοποιεί αντικατάσταση sprite με φόντο 2-bit (4 χρωμάτων) και κίνηση μελάνης με χρήση RINK. Τα πλοία σε μια σειρά κατασκευάζονται με την τοποθέτησή τους χρησιμοποιώντας τη βελτιωμένη εντολή ROUTESP, που είναι διαθέσιμη στην V35.

Παρόλο που το παιχνίδι προσομοιώνει 3 διαστάσεις, δεν χρησιμοποιεί ψευδο-3D

προβολή, αλλά sprite μονοπάτια στα οποία ο εχθρός αλλάζει σε μεγαλύτερη εκδοχή για να δώσει την εντύπωση ότι πλησιάζει. Έτσι, μια σύγκρουση με έναν απομακρυσμένο εχθρό δεν μας επηρεάζει,

ένας αχρησιμοποίητος καταχωρητής κατάστασης σημαίας χρησιμοποιείται για να χαρακτηρίζει τους απομακρυσμένους εχθρούς ως ακίνδυνους.

Στην τρίτη φάση, χρησιμοποιήθηκε σχετική οριζόντια κίνηση των λίθων στο έδαφος σε συνδυασμό με μια επιταχυνόμενη διαδρομή κατακόρυφης κίνησης.



24.7 Αιώνιος βατραχάνθρωπος

Το παιχνίδι "Frogger Eternal" έχει κατασκευαστεί με την έκδοση V36 του 8BP και ο τίτλος του παραπέμπει τόσο στο κλασικό "frogger" της konami που κυκλοφόρησε το 1981 όσο και στην έκθεση "Amstrad Eternal" που πραγματοποιήθηκε το 2019, την εκδήλωση στην οποία έκανε την εμφάνισή του αυτό το παιχνίδι.

Πρόκειται για ένα παιχνίδι προγραμματισμένο σε λειτουργία 1, με χρήση LAYOUT, διαφανή εκτύπωση στο βάτραχο και διαδρομές διαφορετικής φύσης για τα sprites. Μερικά από τα sprites είναι αόρατα, αλλά συγκρουόμενα, όπως 4 "αόρατα" ποτάμια κάτω από κορμούς, νούφαρα και χελώνες που ο βάτραχος πρέπει να πηδήξει, όπως και "αόρατα τείχη" στις πλευρές του ποταμού, ώστε ο βάτραχος να μην μπορεί να ξεφύγει.



24.8 Eridu: Το διαστημικό λιμάνι

Αυτό το παιχνίδι είναι ένας κλώνος του κλασικού "Scramble" της Konami που δημιουργήθηκε το 1981. Στην πραγματικότητα έχει πολλές διαφορές από το πρωτότυπο, αλλά στην ουσία είναι εμπνευσμένο από το κλασικό παιχνίδι, καθώς ενσωματώνει την ανάγκη ανεφοδιασμού, αναγκάζοντας τον παίκτη να πάρει ρίσκα για να καταστρέψει τις δεξαμενές καυσίμων, ώστε να μην χάσει μια ζωή.

Είναι αρκετά γρήγορο παρά το γεγονός ότι τρέχει με έναν ισχυρό scroller, εμφανίζοντας 32 sprites στην οθόνη σε πολλά σημεία. Φτάνει μέχρι και 18 fps.

Το παιχνίδι έχει 5 στάδια και διαφορετική μουσική στο παιχνίδι, καθώς και μια πολύ φανταχτερή γραφική παρουσίαση.

Το Eridu είναι ένα βιντεοπαιχνίδι που συνδέεται με μια αρχαία "απαγορευμένη" ιστορία της ανθρωπότητας. Το Eridu ήταν η πρώτη πόλη του κόσμου, που δημιουργήθηκε από τους "Anunnaki" πριν από 400.000 χρόνια, μια εξωγήινη φυλή σύμφωνα με τις σουμεριακές πινακίδες που βρέθηκαν στην έρημο του Ιράκ. Εκεί δημιουργήσαν ένα



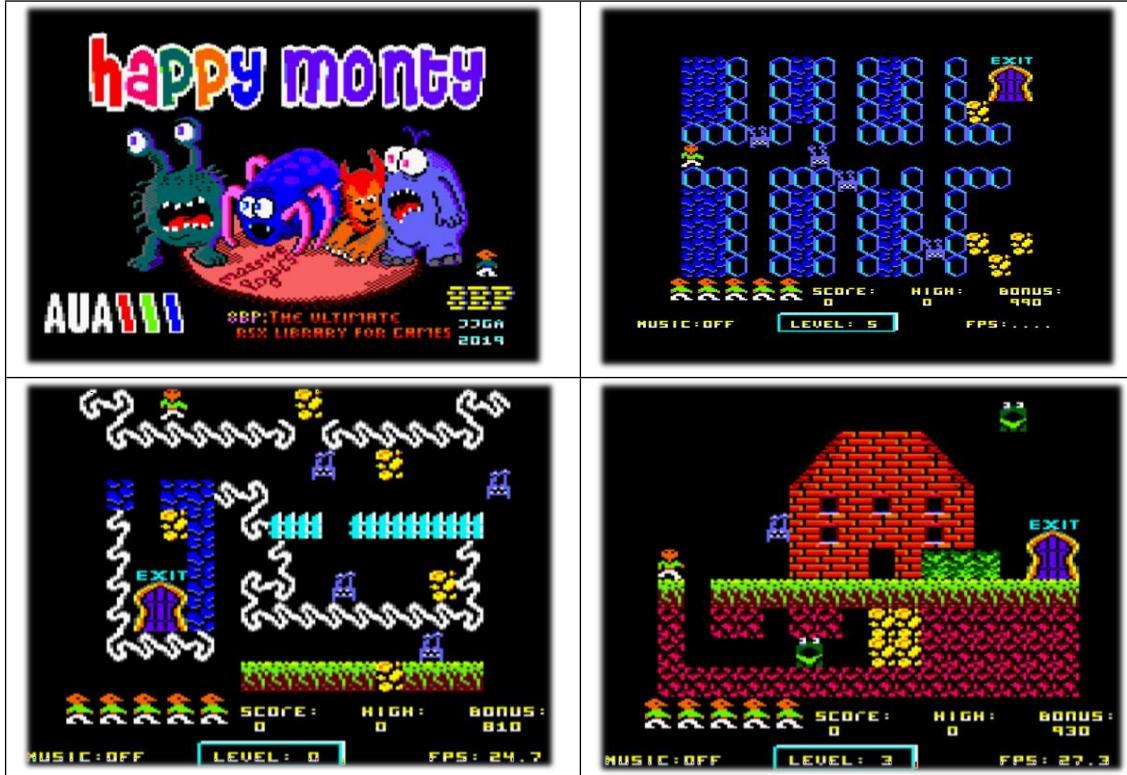
Οι χάρτες των διαφόρων φάσεων φορτώνονται σε διαφορετικές τράπεζες μνήμης, κάθε μία από τις οποίες καταλαμβάνει 500 bytes δεδομένων του κόσμου και 200 bytes για

την περιγραφή των θέσεων των εχθρών. Η κύλιση γίνεται λογικά με την εντολή **|MAP2SP**.

24.9 Happy Monty

Πρόκειται για ένα γρήγορο παιχνίδι που σαρώνει την οθόνη, το οποίο μιμείται σχεδόν τέλεια το κλασικό Mutant Monty. "Κλωνοποιεί" ακόμη και την αρχική του οθόνη. Αυτό το παιχνίδι είναι φτιαγμένο με την έκδοση 37 της βιβλιοθήκης και φτάνει τα 25 FPS.

Κάνει εντατική χρήση της διάταξης και, φυσικά, της μαζικής λογικής. Καταφέρνει να αποθηκεύσει 25 επίπεδα χρησιμοποιώντας μια απλή τεχνική για τη συμπίεση των διατάξεων (κάθε διάταξη δαπανά μόνο 160 bytes) που εξηγείται σε αυτό το βιβλίο.



Μια πρωτότυπη τεχνική που χρησιμοποιείται σε αυτό το παιχνίδι επιτρέπει στους εχθρούς να αλλάζουν διαδρομές. Πρόκειται για αόρατα sprites "αντιστροφής". Όταν ένας εχθρός συγκρούεται με ένα sprite αντιστροφής, αλλάζει τη διαδρομή του και του ανατίθεται η αντίθετη διαδρομή ή ακόμη και μια κάθετη διαδρομή, με αποτέλεσμα να είναι δυνατή η δημιουργία διαδρομών οποιουδήποτε μήκους χωρίς να ορίζονται περισσότερες από μία κάθετη και μία οριζόντια διαδρομή. Μπορούν να δημιουργηθούν ακόμη και βρόχοι.

Αυτό απλοποιεί επίσης τη δημιουργία του χάρτη (διάταξη + εχθροί) κάθε επιπέδου, επειδή όταν εντοπίζετε έναν εχθρό αρκεί να χρησιμοποιήσετε έναν κωδικό (έναν χαρακτήρα) που υποδεικνύει την ταχύτητα και την κατεύθυνση του εχθρού (6 γράμματα χρησιμοποιούνται για όλους τους τύπους εχθρών και περιοδικά αλλάζει η "παλέτα" των εχθρών, οι κούκλες που σχετίζονται με αυτά τα 6 γράμματα).

24.10 Πιλότος Blaster

Πρόκειται για ένα παιχνίδι που δημιουργήθηκε με την έκδοση v39 της 8BP, στην οποία είναι δυνατό να υπάρχουν ηχητικά εφέ (που δημιουργούνται με την εντολή SOUND) ταυτόχρονα με την αναπαραγωγή μουσικής με την εντολή MUSIC. Στην περίπτωση αυτή η εντολή SOUND χρησιμοποιείται για πυροβολισμούς και εκρήξεις και χρησιμοποιεί το τρίτο κανάλι, ενώ η μουσική χρησιμοποιεί τα δύο πρώτα κανάλια.



Το παιχνίδι έχει κύλιση πολλαπλών κατευθύνσεων και είναι εμπνευσμένο από το στυλ παιχνιδιών όπως το "Time Pilot" ή το "Asteroids". Δίπλα στον πίνακα αποτελεσμάτων και ζωών υπάρχει ένα ραντάρ με το οποίο μπορείτε να προσανατολιστείτε στο διάστημα για να εντοπίσετε 3 χαμένους ηθοποιούς που πρέπει να διασώσετε. Το παιχνίδι παρουσιάζει ενδιαφέρον ως προς τον τρόπο με τον οποίο αντιμετωπίζεται όσο το δυνατόν πιο αποτελεσματικά ο προγραμματισμός της κίνησης του σκάφους σε 12 πιθανές κατευθύνσεις.

Εκτός από τα 6 επίπεδα, διαθέτει μια φάση μπόνους στο τέλος κάθε επιπέδου, η οποία σας επιτρέπει να συγκεντρώσετε πόντους και να ανακτήσετε μια ζωή.

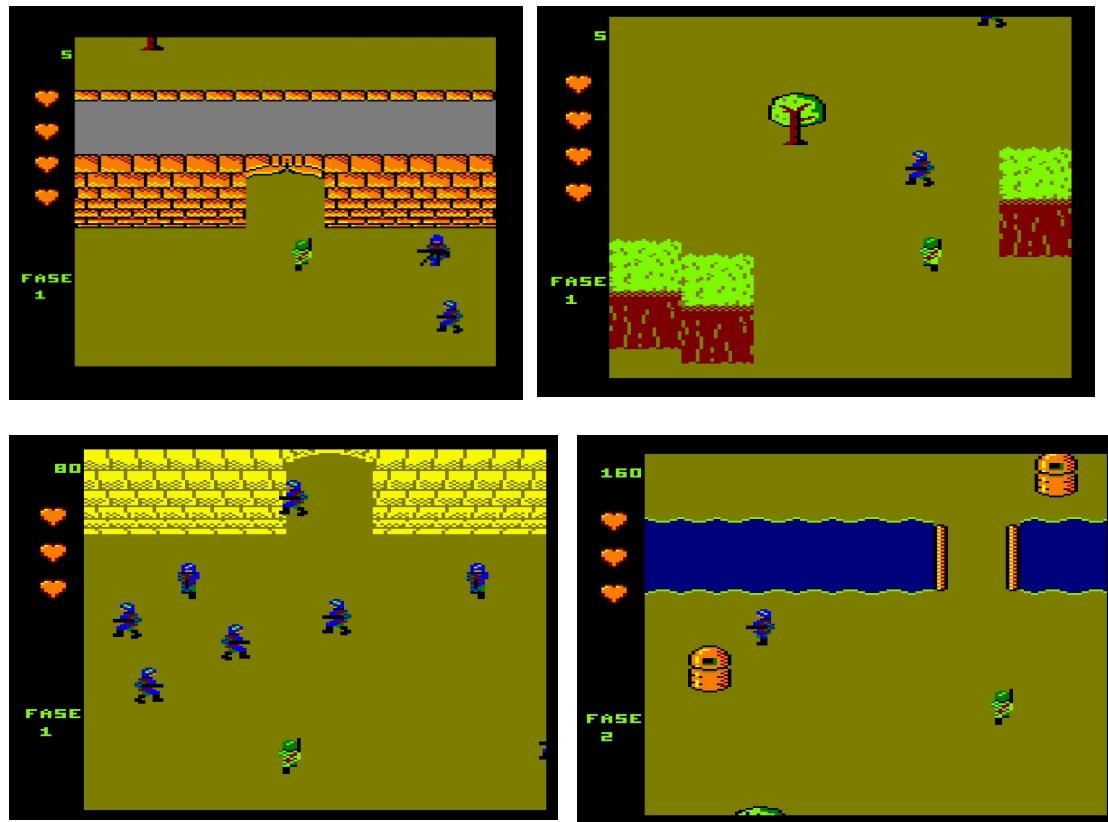
24.11 NOMWARS

Πρόκειται για ένα παιχνίδι στο στυλ του κλασικού "Commando" που δημιουργήθηκε το 1985 από την Capcom. Ένα κλασικό shoot 'em up με κάθετη κύλιση.

Το παιχνίδι αποτελείται από 4 στάδια και μια "φανταχτερή" εισαγωγή, με μια ιστορία για τον πόλεμο της νέας παγκόσμιας τάξης. Έχει δημοσιευτεί σε μορφή DES.

Αυτή η έκδοση εκμεταλλεύεται τη δυνατότητα κύλισης του 8BP (εντολή MAP2SP) και περιλαμβάνει τη διάσημη γέφυρα από την οποία περνάει ο Joe χρησιμοποιώντας μια τεχνική που βασίζεται στην εντολή SETLIMITS του 8BP.

Το παιχνίδι προσφέρεται σε δύο εκδόσεις: την καθαρή έκδοση BASIC και την έκδοση με μεταγλωτισμένο κύκλο (ο κύκλος μεταφράζεται σε γλώσσα C χρησιμοποιώντας το περιτύλιγμα 8BP και το 8BP minibasic). Και οι δύο εκδόσεις είναι πανομοιότυπες, καθώς η μετάφραση σε C είναι ένα πλήρες αντίγραφο, σχεδόν καθρέφτης της έκδοσης BASIC. Στην πραγματικότητα, το παιχνίδι έχει προγραμματιστεί σε BASIC και την τελευταία ημέρα έχει μεταφραστεί σε C με τη δυνατότητα που έχει η 8BP γι' αυτό.

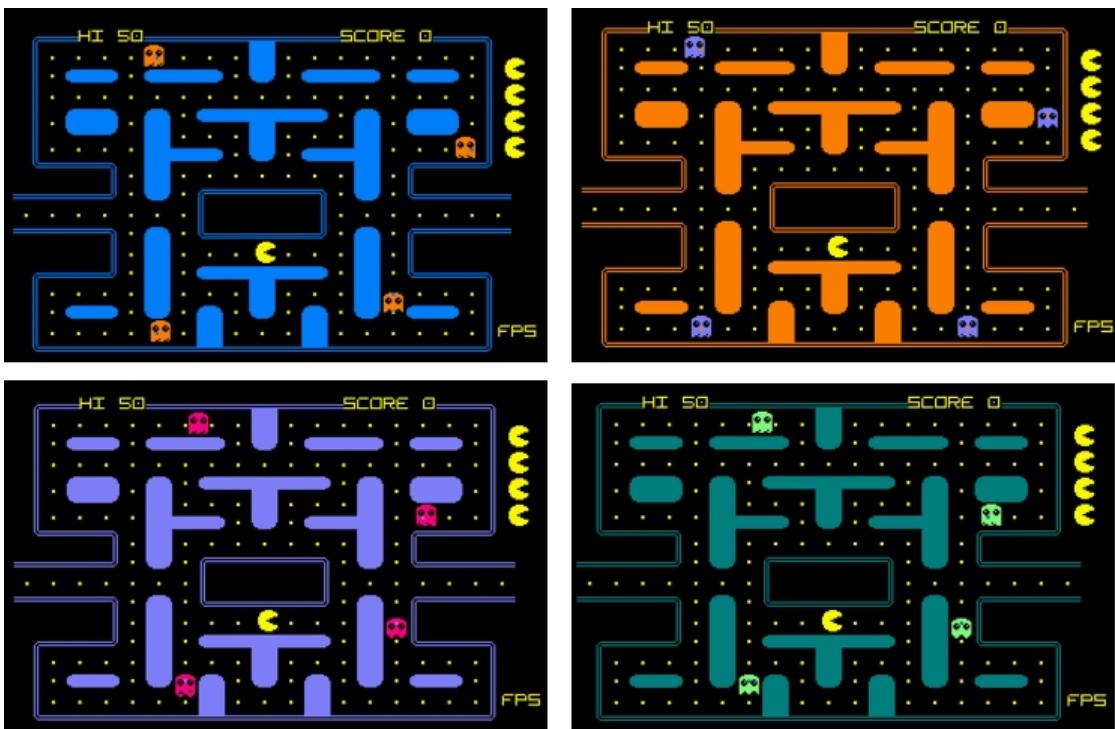


24.12 Paco, o áνθρωπος

Ένα παιχνίδι στο πιο καθαρό στυλ του Pac-Man. Το παιχνίδι περιλαμβάνει δύο φάσεις σε BASIC και δύο με μεταγλωττισμένο κύκλο. Σε BASIC φτάνει τα 19 FPS με λαβύρινθο, συγκρούσεις και λογική 4 φαντασμάτων και σε C φτάνει τα 33 FPS.



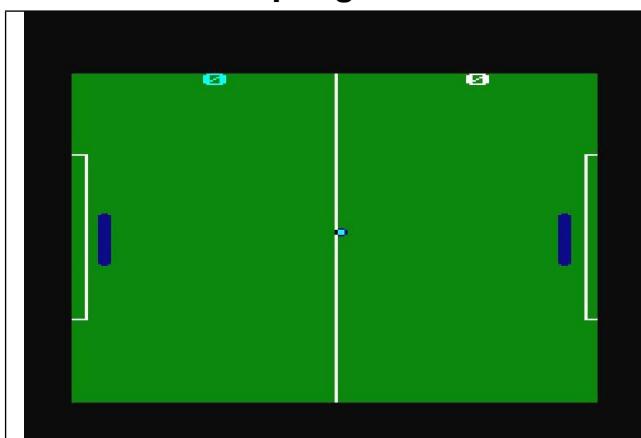
Κάθε επίπεδο Paco αναγνωρίζεται από τα χρώματά του. Τα δύο πρώτα τρέχουν σε BASIC και χρησιμοποιούν "προυπολογισμένες" αποφάσεις φαντασμάτων για να επιτύχουν 19 fps. Το τρίτο και το τέταρτο στάδιο χρησιμοποιούν C και φτάνουν τα 33 fps.



24.13 Μίνι παιχνίδια

Πρόκειται για παιχνίδια που προορίζονται για εκπαιδευτικούς σκοπούς. Απλά κατανοητά και σύντομα, για να βοηθήσουν τους αρχάριους προγραμματιστές στην ανάπτυξη των δικών τους παιχνιδιών.

24.13.1 Mini-pong



Εικ. 123 Βιντεοπαιχνίδι "Mini-pong"

Πρόκειται για ένα πολύ απλό και διδακτικό βιντεοπαιχνίδι. Βασίζεται στο κλασικό "Pong" της Atari (1972).

Η μπάρα του αντιπάλου (ο υπολογιστής) αρχίζει να παίρνει αποφάσεις όταν η μπάλα περάσει το μισό γήπεδο, οπότε είναι δυνατόν να τον νικήσετε. Αν τον βάλουμε να παίρνει αποφάσεις νωρίτερα, έρχεται μια στιγμή που είναι αδύνατο να νικήσει.

Μερικές σύντομες λεπτομέρειες για το παιχνίδι:

- Χρήση του |COLSPALL: για την ανίχνευση συγκρούσεων μεταξύ του "συγκρουστή" (της μπάλας) και των "συγκρουστών" (των ράβδων). Στο status byte των sprites, η σημαία collider είναι ενεργοποιημένη για την μπάλα (sprite 29) και η σημαία collider είναι ενεργοποιημένη στις 31 (η δική μας μπάρα) και 30 (η μπάρα του αντιπάλου).
- Χρησιμοποιήστε το overwrite στη μπάλα για να σεβαστείτε τη λευκή λωρίδα στο γήπεδο και να μην τη σβήσετε όταν περνάτε. Για να το κάνετε αυτό, χρησιμοποιήστε μια παλέτα με overwrite και ενεργοποιήστε τη σημαία overwrite στο status byte του sprite της μπάλας (το 29o).
- υπάρχουν μόνο δύο εικόνες (η μπάλα=17 και η μπάρα=16) που αντιστοιχίζονται στα 2 sprites (στα sprites 30 και 31 αντιστοιχίζεται η εικόνα 16 και στο sprite 29 η εικόνα 17).
- Τα sprites χρησιμοποιούν αυτόματη κίνηση. Για το σκοπό αυτό έχουν ενεργοποιημένη τη σημαία αυτόματης κίνησης και η εντολή |AUTOALL τα μετακινεί (αλλάζει τις συντεταγμένες τους) ανάλογα με την ταχύτητά τους.
- Όλα τα sprites εκτυπώνονται με |PRINTSPALL σε κάθε κύκλο παιχνιδιού.

24.13.2 Mini-Invaders

Όπως και το "Mini-pong", πρόκειται για ένα παιχνίδι που δημιουργήθηκε για εκπαιδευτικούς σκοπούς, εμπνευσμένο από το κλασικό "Space Invaders" της Taito (1978).



Εικ. 124 Βιντεοπαιχνίδι "μίνι εισβολείς".

Μερικές συμβουλές για το πώς γίνεται αυτό:

- Το παιχνίδι χρησιμοποιεί 32 sprites
- Το πλοίο είναι το sprite 31
- Οι βολές που μπορείτε να ρίξετε με το σκάφος είναι 29 και 30.
- Οι εισβολείς πυροβολούν χρησιμοποιώντας sprite 28
- Οι εισβολείς χρησιμοποιούν τα sprites 0 έως 27 (συνολικά 28 εισβολείς).
- Τα sprites 31, 30 και 29 έχουν ενεργή σημαία συγκρούσεων.
- Τα υπόλοιπα sprites "συγκρούονται" και έχουν ενεργή σημαία σύγκρουσης.
- Οι εισβολείς έχουν μια ενεργή σημαία αυτόματης κίνησης και συνδέονται με τη διαδρομή "0" που τους μετακινεί από δεξιά προς αριστερά και προς τα κάτω, τυπικά για τους εισβολείς.
- Οι ενεργοποιητές πλοίων και εισβολέων χρησιμοποιούν ένα χαρακτηριστικό του

V27, διατρέχουν την οθόνη και κατά την έξοδο απενεργοποιούνται αυτόματα με μια καθορισμένη αλλαγή κατάστασης στο τέλος της διαδρομής τους, απλοποιώντας έτσι τη λογική της BASIC και επιταχύνοντας έτσι το παιχνίδι.

25 ΠΑΡΑΡΤΗΜΑ I: Οργάνωση μνήμης βίντεο

25.1 Το ανθρώπινο μάτι και η ανάλυση του CPC

Η μνήμη βίντεο του Amstrad CPC έχει 3 τρόπους λειτουργίας. Ο πιο συχνά χρησιμοποιούμενος τρόπος λειτουργίας για παιχνίδια είναι ο τρόπος λειτουργίας 0 (160x200) επειδή έχει περισσότερα χρώματα, αλλά ο τρόπος λειτουργίας 1 (320x200) έχει επίσης χρησιμοποιηθεί πολύ για τον προγραμματισμό παιχνιδιών. Ο τρόπος λειτουργίας 2 (640x200) χρησιμοποιήθηκε σπάνια ή ποτέ για παιχνίδια λόγω των περιορισμένων 2 χρωμάτων του.

Δεδομένου ότι το ποσό της μνήμης βίντεο είναι το ίδιο, η ανάλυση θυσιάζεται για να κερδηθεί το ποσό των χρωμάτων, αλλά περιέργως, οριζόντια, που είναι η μεγαλύτερη πλευρά της οθόνης, υπάρχει μικρότερη ανάλυση από ό,τι κάθετα (160 οριζόντια και 200 κάθετα). Ισως αναρωτηθείτε γιατί. Άλλωστε, δεν είναι ο μόνος μικροϋπολογιστής που το έκανε αυτό, πολλοί άλλοι υπολογιστές χρησιμοποίησαν επίσης την ίδια στρατηγική με την οριζόντια πλευρά.

Ο λόγος έχει να κάνει με τη λειτουργία του ανθρώπινου οπτικού συστήματος. Το μάτι αντιλαμβάνεται περισσότερες λεπτομέρειες κάθετα απ' ό,τι οριζόντια, οπότε η "καταστροφή" της ανάλυσης στον οριζόντιο άξονα δεν είναι τόσο σοβαρή όσο η καταστροφή της κάθετα. Υποκειμενικά το αποτέλεσμα είναι πιο αποδεκτό. Το ανθρώπινο οπτικό σύστημα είναι σαν τη λειτουργία 0, πολύ ευρεία εικονοστοιχεία. Γι' αυτό και το οριζόντιο οπτικό πεδίο είναι μεγαλύτερο από το κάθετο.

25.2 Μνήμη βίντεο

Οι πιο πλήρεις και σαφείς πληροφορίες μπορούν να βρεθούν στο εγχειρίδιο firmware της Amstrad. Αυτές οι πληροφορίες θα σας φανούν χρήσιμες αν θέλετε να φτιάξετε έναν βελτιωμένο επεξεργαστή sprite ή αν θέλετε να μπείτε σε assembler και να προγραμματίσετε ρουτίνες αντικατάστασης εκτυπώσεων ή οτιδήποτε άλλο.

25.2.1 Λειτουργία 2

Στη λειτουργία 2, κάθε εικονοστοιχείο αντιπροσωπεύεται από ένα bit. Έτσι, ένα byte αντιπροσωπεύει 8 pixel. Εάν πάρουμε οποιοδήποτε byte από τη μνήμη βίντεο, η αντιστοιχία του με τα εικονοστοιχεία είναι 1 bit για κάθε εικονοστοιχείο, σε αυτόν τον πίνακα αναπαρίστανται τα bit και σε ποια εικονοστοιχεία (p) αντιστοιχούν.

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
p0	p1	p2	p3	p4	p5	p6	p7

Σε ένα byte, το bit 7 αριθμείται ως το πιο αριστερό bit. Το εικονοστοιχείο 0 είναι επίσης το πιο αριστερό εικονοστοιχείο, δηλαδή δεν υπάρχει τίποτα "ανάποδο" εδώ. Όλα είναι σωστά.

25.2.2 Λειτουργία 1

Στη λειτουργία 1 έχουμε 4 χρώματα (που αντιπροσωπεύονται από 2 bit). Συνεπώς, ένα byte αντιπροσωπεύει 4 εικονοστοιχεία. Η αντιστοιχία μεταξύ εικονοστοιχείων και bit είναι κάπως πιο περίπλοκη. Το εικονοστοιχείο 0 για παράδειγμα κωδικοποιείται με τα

bit 7 και 3.

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
p0(0)	p1(0)	p2(0)	p3(0)	p0(1)	p1(1)	p2(1)	p3(1)

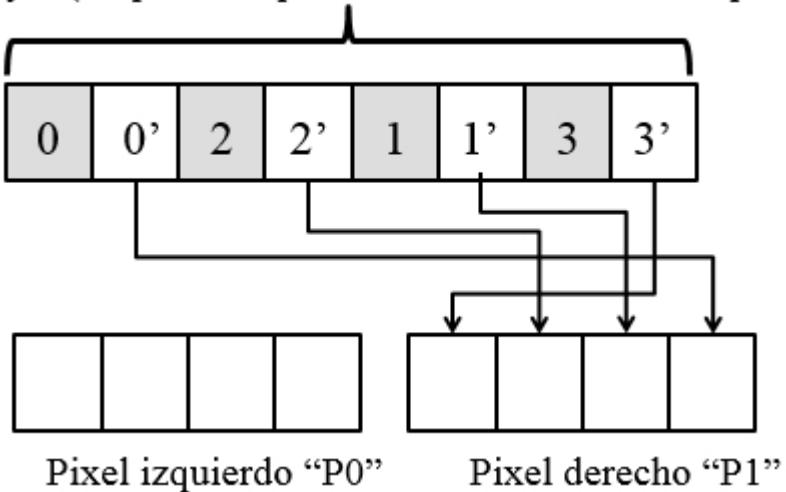
25.2.3 Λειτουργία 0

Εδώ έχουμε ένα μικρό χάος. Κάθε byte αντιπροσωπεύει μόνο δύο pixels, των οποίων η αντιστοιχία με τα bits του byte έχει ως εξής: Το εικονοστοιχείο 0 κωδικοποιείται με τα bit 7,5,3 και 1.

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
p0(0)	p1(0)	p0(2)	p1(2)	p0(1)	p1(1)	p0(3)	p1(3)

Η παρακάτω εικόνα θα σας το κάνει σίγουρα πιο σαφές:

Byte (lo que se imprime en las direcciones de pantalla)



Σχ. 125 εικονοστοιχεία και bits στη λειτουργία 0

Δεν μπορώ να σας πω ποιος είναι ο σκοτεινός λόγος που η μνήμη έχει οργανωθεί με αυτόν τον τρόπο, αλλά φαντάζομαι ότι η αιτία βρίσκεται στο GATE ARRAY, το τσιπ που μεταφράζει αυτά τα bits σε σήμα βίντεο. Φαντάζομαι ότι ο σχεδιαστής μείωσε το κύκλωμα με αυτό το διεστραμμένο σχέδιο.

25.2.4 Μνήμη οθόνης

Τα εικονοστοιχεία της οθόνης που ανήκουν στην ίδια γραμμή κωδικοποιούνται σε bytes που είναι επίσης συνεχόμενα. Ωστόσο, από τη μία γραμμή στην άλλη υπάρχουν άλματα.

Αν προχωρήσουμε προς τα εμπρός στις διευθύνσεις μνήμης, όταν φτάσουμε στο τέλος μιας γραμμής μεταπηδάμε σε μια γραμμή που βρίσκεται 8 γραμμές πιο κάτω. Και αν θέλουμε να συνεχίσουμε στην επόμενη γραμμή, πρέπει να κάνουμε άλμα σε διευθύνσεις μνήμης σε 2048 θέσεις.

Ο παρακάτω πίνακας παρουσιάζει τη μνήμη βίντεο. Στα αριστερά βρίσκεται η γραμμή των χαρακτήρων (από 1 έως 25) και για κάθε γραμμή, η αρχική διεύθυνση κάθε μίας από τις 8 γραμμές σάρωσης που την αποτελούν (με την ονομασία ROW0 ...ROW7).

CHARACTER LINE	R0W0	R0W1	R0W2	R0W3	R0W4	R0W5	R0W6	R0W7	C001	C002
1	C000	C800	D000	D800	E000	E800	F000	F800	C000	
2	C050	C850	D050	D850	E050	E850	F050	F850	C800	
3	C0A0	C8A0	D0A0	D8A0	E0A0	E8A0	F0A0	F8A0	D000	
4	C0F0	C8F0	D0F0	D8F0	E0F0	E8F0	F0F0	F8F0	D800	
5	C140	C940	D140	D940	E140	E940	F140	F940	E000	
6	C190	C990	D190	D990	E190	E990	F190	F990	E800	
7	C1E0	C9E0	D1E0	D9E0	E1E0	E9E0	F1E0	F9E0	F000	
8	C230	CA30	D230	DA30	E230	EA30	F230	FA30	F800	
9	C280	CA80	D280	DA80	E280	EA80	F280	FA80		
10	C2D0	CAD0	D2D0	DAD0	E2D0	EAD0	F2D0	FAD0		
11	C320	CB20	D320	DB20	E320	EB20	F320	FB20		
12	C370	CB70	D370	DB70	E370	EB70	F370	FB70		
13	C3C0	CBC0	D3C0	DBC0	E3C0	EBC0	F3C0	FBC0		
14	C410	CC10	D410	DC10	E410	EC10	F410	FC10		
15	C460	CC60	D460	DC60	E460	EC60	F460	FC60		
16	C4B0	CCB0	D4B0	DCB0	E4B0	ECB0	F4B0	FCB0		
17	C500	CD00	D500	DD00	E500	ED00	F500	FD00		
18	C550	CD50	D550	DD50	E550	ED50	F550	FD50		
19	C5A0	CDA0	D5A0	DDA0	E5A0	EDA0	F5A0	FDA0		
20	C5F0	CDF0	D5F0	DDF0	E5F0	ED50	F550	FD50		
21	C640	CE40	D640	DE40	E640	EE40	F640	FE40		
22	C690	CE90	D690	DE90	E690	EE90	F690	FE90		
23	C6E0	CEE0	D6E0	DEE0	E6E0	EEE0	F6E0	FEE0		
24	C730	CF30	D730	DF30	E730	EF30	F730	FF30		
25	C780	CF80	D780	DF80	E780	EF80	F780	FF80		
spare start	C7D0	CFD0	D7D0	DFD0	E7D0	EFD0	F7D0	FFD0		
spare end	C7FF	CFFF	D7FF	DFFF	E7FF	EFFF	F7FF	FFFF		

Εικ. 126 Χάρτης μνήμης οθόνης

*Direcciones de la
esquina superior
izquierda de la
pantalla*

C000 = comienzo de pantalla
= 49152 , es decir 48KB

FFFF= fin de pantalla
= 65535

La pantalla mide:
65535 – 49152 = 16384 =16KB

Η οθόνη της Amstrad έχει 200 γραμμές x 80 bytes πλάτος η κάθε μία, οπότε η μνήμη της οθόνης που εμφανίζεται είναι $200 \times 80 = 16.000$ bytes. Ωστόσο, η μνήμη βίντεο είναι 16384 bytes. Υπάρχουν 384 bytes "κρυμμένα" σε 8 τμήματα των 48 bytes το καθένα, τα οποία δεν εμφανίζονται στην οθόνη, παρόλο που αποτελούν μέρος της μνήμης βίντεο. Αυτά τα 8 τμήματα ονομάζονται "εφεδρικά" στον παραπάνω πίνακα. Κάθε τμήμα έχει μήκος 48 bytes επειδή, όπως είπα και πριν, για να μεταπροσετε από τη μία γραμμή στην επόμενη πρέπει να προσθέσετε 2048 bytes, αλλά στην πραγματικότητα οι 25 συνεχόμενες γραμμές μνήμης που τα χωρίζουν καταλαμβάνουν μόνο 25×80 bytes = 2000 bytes.

Από &C7D0 έως C7FF και τα δύο συμπεριλαμβανομένων Από &CFD0 έως CFFF και τα δύο συμπεριλαμβανομένων Από &D7D0 έως D7FF και τα δύο συμπεριλαμβανομένων Από &DFD0 έως DFFF και τα δύο συμπεριλαμβανομένων Από &E7D0 έως E7FF και τα δύο συμπεριλαμβανομένων Από &EFDO έως EFFF και τα δύο συμπεριλαμβανομένων Από &F7D0 έως F7FF και τα δύο συμπεριλαμβανομένων Από &FFD0 έως F7FF και τα δύο συμπεριλαμβανομένων Από &FFF0 έως FFFF και τα δύο συμπεριλαμβανομένων

Μπορείτε να το ελέγξετε αυτό κάνοντας POKE σε αυτές τις διευθύνσεις μνήμης και θα δείτε ότι δεν θα αλλάξει το περιεχόμενο της οθόνης.

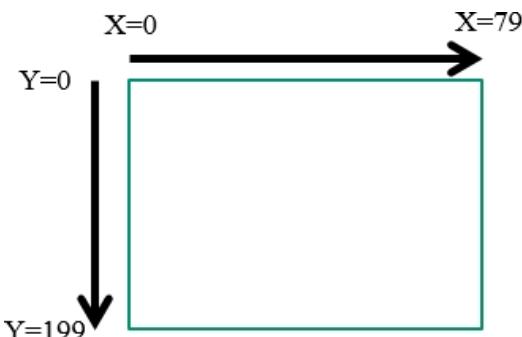
Είναι δελεαστικό να σκεφτεί κανείς να χρησιμοποιήσει αυτές τις "κρυφές" περιοχές μνήμης για την αποθήκευση μικρών ρουτινών συναρμολόγησης ή μεταβλητών. Ωστόσο, αυτό είναι επικίνδυνο επειδή μια

Η εντολή MODE που εκτελείται από τη BASIC διαγράφει πλήρως αυτά τα τμήματα μνήμης, οπότε αν τη χρησιμοποιήσετε θα πρέπει να το γνωρίζετε αυτό. Στη βιβλιοθήκη 8BP τα τμήματα αυτά χρησιμοποιούνται για την αποθήκευση τοπικών μεταβλητών ορισμένων συναρτήσεων, των οποίων η τιμή μπορεί να διαγραφεί με ασφάλεια.

25.3 Υπολογισμός διεύθυνσης οθόνης

Αν θέλετε να μάθετε τη διεύθυνση μνήμης στην οποία αντιστοιχούν συγκεκριμένες συντεταγμένες 8BP, θα πρέπει να εκτελέσετε την ακόλουθη ενέργεια

$$\text{Dir} = \&C000 + \text{INT}(y/8)*80 + (y \bmod 8)*2048 + x$$



Σχ. 127 Συντεταγμένες οθόνης σε 8BP

Αυτό είναι πολύ χρήσιμο αν θέλετε για παράδειγμα να χρησιμοποιήσετε το PEEK για να διαπιστώσετε αν υπάρχει ένα συγκεκριμένο στοιχείο ή χρώμα σε ένα συγκεκριμένο byte στην οθόνη και να το χρησιμοποιήσετε ως μηχανισμό ανίχνευσης συγκρούσεων. Αυτή η τεχνική χρησιμοποιείται στο βιντεοπαιχνίδι "3D-Racing One".

Σε περίπτωση που θέλετε να μάθετε την κατεύθυνση κάποιων γραφικών συντεταγμένων (αυτές που χρησιμοποιούνται από την εντολή BASIC PLOT) πρέπει πρώτα να κάνετε $y2=(200-y)/2$, $x2=x/8$

$$\text{Dir} = \&C000 + \text{INT}(y2/8)*80 + (y2 \bmod 8)*2048 + x2$$

25.4 Σαρώσεις οθόνης

Η Amstrad παράγει 50 καρέ ανά δευτερόλεπτο. Αυτό σημαίνει ότι περίπου κάθε 20ms πρέπει να παράγεται μια νέα σάρωση οθόνης.

Θα μπορούσατε να σκεφτείτε ότι ίσως το σύρσιμο της οθόνης, το οποίο ζωγραφίζει την οθόνη, καταναλώνει ένα κλάσμα αυτών των 20ms, αλλά δεν συμβαίνει αυτό. Η ζωγραφική της οθόνης παίρνει το Amstrad όλα τα 20ms, οπότε ακόμα και αν συγχρονίσετε την εκτύπωση του sprite σας με το σκούπισμα της οθόνης είναι πολύ πιθανό να σας προλάβει, προκαλώντας δύο γνωστά φαινόμενα:

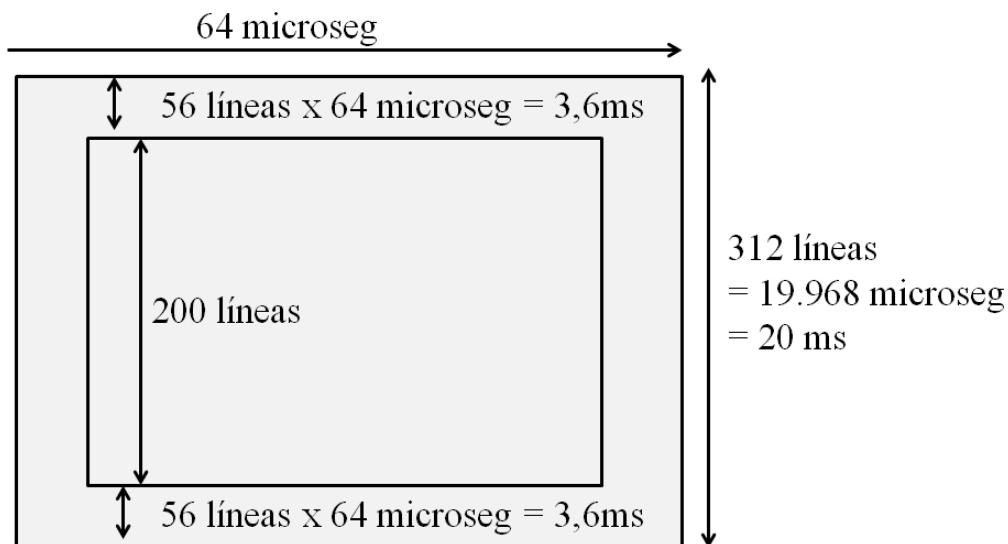
- **Το τρεμόπαιγμα** εμφανίζεται όταν διαγράφετε το sprite πριν το εκτυπώσετε στη νέα του θέση. Για να το αποφύγετε αυτό υπάρχει μια πολύ απλή λύση: μην το διαγράψετε. Απλά κάντε το sprite να διαγράψει το ίδιο του το ίχνος, αφήνοντας ένα περίγραμμα στο sprite για να εκπληρώσει αυτή τη λειτουργία. Το sprite είναι μεγαλύτερο, αλλά δεν θα τρεμοπαίζει, ακόμα και αν πιαστεί στη μέση, επειδή δεν εξαφανίζεται.

- **Σκίσιμο:** συμβαίνει όταν μας πιάνει το σκούπισμα στη μέση του sprite. Το μισό εκτυπώνεται με τη νέα θέση (κεφάλι και κορμός) και το άλλο μισό όχι.

δίνει χρόνο (τα πόδια). Τότε το sprite εκτυπώνεται "λάθος", αν και διορθώνεται στο επόμενο καρέ, αλλά για μια στιγμή είναι σαν να είναι παραμορφωμένο ή σπασμένο. Το σκίσιμο είναι ένα κακό εφέ, αλλά πολύ πιο αποδεκτό από το τρεμόπαιγμα. Η τέλεια λύση είναι να ελέγχουμε κάθε χιλιοστό του δευτερολέπτου πού είναι το τρεμόπαιγμα, ώστε να εκτυπώνεται κάθε sprite χωρίς να φτάνει σε εμάς.

Μια τυπική σύσταση είναι να εκτυπώνετε τα sprites από κάτω προς τα πάνω, για να ελαχιστοποιήσετε αυτά τα φαινόμενα. Με αυτόν τον τρόπο είναι δυνατόν να λάβετε τη σάρωση μόνο μία φορά σε ένα από τα sprites, ενώ εκτυπώνοντας από πάνω προς τα κάτω, μπορείτε να λάβετε τη σάρωση σε πολλά sprites, επειδή και τα δύο (CPU και καθοδικές ακτίνες) λειτουργούν προς την ίδια κατεύθυνση. Δυστυχώς, το πιο ενδιαφέρον πράγμα που μπορείτε να κάνετε είναι να ταξινομήσετε από πάνω προς τα κάτω για να δώσετε εφέ βάθους στα sprites (χρήσιμο σε ορισμένα παιχνίδια όπως το "Golden axe", το "Double dragon", το "Renegade" κ.λπ.)

Οι χρόνοι που καταναλώνει η οθόνη είναι οι εξής. Σημειώστε ότι από τη στιγμή της διακοπής της σάρωσης, έχετε 3,5ms για να ζωγραφίσετε χωρίς να είναι δυνατόν να πιαστείτε. Αλλά σε αυτό το χρονικό διάστημα μπορείτε να εκτυπώσετε το πολύ 2 μικρά sprites.



Εικ. 128 φορές σε μια σάρωση οθόνης

25.5 Πώς να φτιάξετε μια οθόνη φόρτωσης για το παιχνίδι σας

Υπάρχουν πολλοί τρόποι για να το κάνετε. Ένας πολύ απλός είναι να φτιάξετε ένα γραφικό με το πρόγραμμα SPEDIT τροποποιημένο ώστε να σας επιτρέπει να ζωγραφίζετε σε όλη την οθόνη χωρίς να εμφανίζετε μενού και στο τέλος να πατήσετε κάποιο πλήκτρο για να εκκινήσετε μια εντολή SAVE όπως αυτή εδώ

SAVE "mipantalla.bin", b, &C000, 16384

Όπως μπορείτε να δείτε, η εντολή αποθηκεύει 16KB από τη διεύθυνση έναρξης της οθόνης, η οποία είναι &C000.

Ο τρόπος φόρτωσης θα ήταν ο εξής

LOAD " mipantalla.bin", &C000

Εάν δεν χρησιμοποιείτε την προεπιλεγμένη παλέτα, τότε πρέπει πρώτα να εκτελέσετε τις εντολές INK που αντιστοιχούν στην παλέτα που έχετε χρησιμοποιήσει, πριν φορτώσετε την οθόνη. Κατά τη φόρτωση της οθόνης, θα δείτε πώς η οθόνη σχεδιάζεται σιγά-σιγά στην οθόνη καθώς φορτώνεται, αφού εκεί ακριβώς τη φορτώνετε, στη μνήμη βίντεο.

Ένας άλλος τρόπος είναι να δημιουργήσετε μια καλά επεξεργασμένη διάταξη και να την αποθηκεύσετε χρησιμοποιώντας την παραπάνω εντολή SAVE. Το θέμα είναι να δημιουργήσετε ένα σχέδιο και όπως μπορείτε να δείτε υπάρχουν πολλοί τρόποι.

Τέλος, μπορείτε να χρησιμοποιήσετε ένα εργαλείο όπως το ConvImgCPC (υπάρχουν και άλλα), έναν μετατροπέα/επεξεργαστή εικόνων που λειτουργεί στα Windows. Αυτό το εργαλείο σας επιτρέπει να μετατρέψετε οποιαδήποτε εικόνα (η οποία μπορεί να είναι η σάρωση ενός σχεδίου σας) σε ένα δυαδικό αρχείο (με επέκταση .scr) κατάλληλο για το CPC. Αυτό το εργαλείο σας επιτρέπει επίσης να την επεξεργαστείτε pixel προς pixel και να τη ρετουσάρετε μέχρι να την κάνετε τέλεια. Μπορείτε ακόμη και να το επεξεργαστείτε από το μηδέν, χωρίς να σαρώσετε τίποτα. Κατά τη γνώμη μου αυτή είναι η καλύτερη επιλογή.

Για να τοποθετήσετε αυτό το αρχείο σε ένα δίσκο (σε ένα αρχείο .dsk) πρέπει να χρησιμοποιήσετε το CPCDiskXP το οποίο είναι ένα άλλο εργαλείο που σας επιτρέπει να τοποθετείτε αρχεία μέσα σε αρχεία .dsk.

Μόλις μπείτε στο .dsk μπορείτε να το φορτώσετε με LOAD "mipantalla.bin",&C000 Ωστόσο, τα χρώματα δεν θα φαίνονται σωστά, επειδή το ConvImgCPC προσαρμόζει την παλέτα ώστε να είναι όσο το δυνατόν πιο κοντά στα αρχικά χρώματα. Για να τα δείτε σωστά πρέπει να εκτελέσετε τη ρουτίνα όπου το ConvImg τοποθετεί τη ρουτίνα αλλαγής παλέτας, η οποία είναι η CALL &C7D0.

Αυτή η ρουτίνα είναι "κρυμμένη" στο πρώτο από τα 8 κρυφά τμήματα της μνήμης βίντεο, οπότε η εικόνα .scr δεν καταλαμβάνει περισσότερο χώρο επειδή περιέχει αυτή τη ρουτίνα. Το κακό είναι ότι μέχρι να μην φορτώσετε την οθόνη δεν μπορείτε να την εκτελέσετε και επομένως θα δείτε πώς φορτώνει την εικόνα με λάθος χρώματα και στο τέλος θα μπορείτε να επικαλεστείτε αυτή την κλήση, αλλάζοντας τα χρώματα. Αυτό που μπορείτε να κάνετε είναι να προετοιμάσετε ένα ειδικό αρχείο παλέτας. Κάνοντας αυτό:

Φόρτωση "image.scr", &c000

Αποθήκευση "palette.bin", b, &c7d0, 48, &c7d0

Τώρα έχετε ένα αρχείο 48 byte που περιέχει την παλέτα. Στον φορτωτή του παιχνιδιού σας θα κάνετε το εξής:

Φόρτωση "!palette.bin"

Καλέστε &c7d0

Φόρτωση " !image.scr", &c000

Συνιστώ να τοποθετήσετε απλώς μερικές εντολές INK πριν από την εντολή LOAD που φορτώνει την εικόνα.

Φόρτωση "image.scr", &c000

Και αμέσως μετά, πριν χρησιμοποιήσετε την 8BP για να εκτυπώσετε sprites κ.λπ., είναι βολικό να διαγράψετε το κρυφό τμήμα όπου η ConvImg αφήνει τη ρουτίνα, καθώς είναι ένας χώρος που χρησιμοποιεί η 8BP για μεταβλητές και αν αυτές δεν είναι αρχικά μηδενισμένες, μπορεί να παρεμβληθούν.

for i = &c000+2000 έως &c000+2000+48: poke i,0:NEXT

με λίγα λόγια:

10 <πολλές εντολές INK>

20 Φόρτωση " !image.scr", &c000

30 for i = &c000+2000 έως &c000+2000+48: poke i,0:NEXT

Για να αφήσετε την παλέτα στις προεπιλεγμένες τιμές της, χρησιμοποιήστε το CALL &BC02, το οποίο είναι μια ρουτίνα του υλικολογισμικού.

Και να αποθηκεύσετε την οθόνη σε μια ταινία;

Είδαμε πώς να το κάνουμε στο δίσκο, αλλά το CPCDiskXP δεν θα αφήσει το αρχείο .scr στην ταινία, οπότε πρέπει να κάνουμε κάτι τέτοιο:

```
|DISC  
MNHMH 15999  
LOAD "image.scr", 16000  
|TAPE  
TAXYTHTA EΓΓΡΑΦΗΣ 1  
SAVE "imagen.scr",b,16000,16384
```

Και όταν το φορτώνουμε, το φορτώνουμε με τον ίδιο τρόπο όπως στο δίσκο:

Φόρτωση " !image.scr", &c000

26 ΠΑΡΑΡΤΗΜΑ II: Η παλέτα

Ο ακόλουθος πίνακας παρουσιάζει την παλέτα AMSTRAD. Μέσα σε κάθε χρώμα και σε παρένθεση είναι ο αριθμός μελανιού που έχει αντιστοιχιστεί στο συγκεκριμένο χρώμα στην προεπιλεγμένη παλέτα. Τα 27 χρώματα είναι τα εξής:

0 - Negro (5)	1 - Azul (0,14)	2 - Azul claro (6)	3 - Rojo	4 - Magenta	5 - Violeta	6 - Rojo claro (3)	7 - Púrpura	8 - Magenta claro (7)
9 - Verde	10 - Cyan (8)	11 - Azul cielo (15)	12 - Amarillo (9)	13 - Gris	14 - Azul pálido (10)	15 - Anaranjado	16 - Rosa (11)	17 - Magenta pálido
18 - Verde claro (12)	19 - Verde mar	20 - Cyan claro (2)	21 - Verde lima	22 - Verde pálido (13)	23 - Cyan pálido	24 - Amarillo claro (1)	25 - Amarillo pálido	26 - Blanco (4)

Οι προεπιλεγμένες τιμές παλέτας σε κάθε λειτουργία είναι:

Modo 2:

0: Azul (paleta 1)	1: Amarillo intenso (paleta 24)
--------------------	---------------------------------

Modo 1:

0: Azul (paleta 1)	1: Amarillo intenso (paleta 24)
2: Cyan claro (paleta 20)	3: Rojo claro (paleta 6)

Modo 0:

0: Azul (paleta 1)	1: Amarillo intenso (paleta 24)	2: Cyan claro (paleta 20)	3: Rojo claro (paleta 6)
4: Blanco (paleta 26)	5: Negro (paleta 0)	6: Azul claro (paleta 2)	7: Magenta claro (paleta 8)
8: Cyan (paleta 10)	9: Amarillo (paleta 12)	10: Azul pálido (paleta 14)	11: Rosa (paleta 16)
12: Verde claro (paleta 18)	13: Verde pálido (paleta 22)	14: Parpadeo Azul/Amarillo	15: Parpadeo azul cielo/Rosa

Οι τιμές της παλέτας σε κάθε λειτουργία διαχειρίζονται με την εντολή INK, δείτε το εγχειρίδιο αναφοράς Amstrad BASIC για περισσότερες πληροφορίες. Για παράδειγμα, για να ορίσουμε το μηδενικό μελάνι ως κόκκινο, συμβουλευόμαστε την παλέτα των 27, βλέπομε ότι το κόκκινο είναι το έκτο χρώμα και γράφουμε

INK 0,6

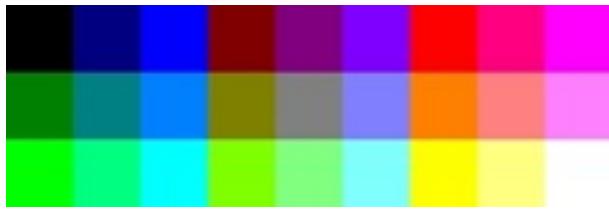
Και έχουμε ήδη ρυθμίσει το μηδενικό μελάνι να είναι κόκκινο. Όπως μπορείτε να δείτε, ένα μελάνι δεν είναι ένα συγκεκριμένο χρώμα, αλλά μπορεί να ρυθμιστεί ώστε να είναι οποιοδήποτε χρώμα θέλετε.

Ο λόγος για τον οποίο η παλέτα Amstrad είναι τόσο καλή είναι ότι τα 27 χρώματα προσφέρουν τόσες πολλές δυνατότητες, παρόλο που μόνο 16 μπορούν να χρησιμοποιηθούν ταυτόχρονα. Τα χρώματα είναι ταξινομημένα με βάση τη φωτεινότητα.

Αν αναπαραστήσουμε τα χρώματα του Amstrad σε κλίμακα RGB 24bit (8 bit ανά στοιχείο), θα διαπιστώσουμε ότι έχει 3 τιμές κόκκινου, 3 πράσινου και 3 μπλε, (οι τιμές αυτές είναι 0,127 και 255) και ο αριθμός των συνδυασμών είναι $3 \times 3 \times 3 = 27$. Το γκρι

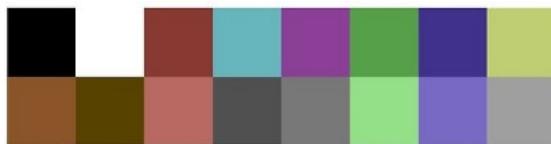
χρώμα βρίσκεται ακριβώς στο κέντρο της παλέτας, όπου οι τιμές των R,G,B είναι ίσες.

(R=127,G=127,B=127). Οι 3 συνιστώσες είναι επίσης ίδιες στο λευκό (R=255,G=255,B=255) και στο μαύρο (R=0,G=0,B=0).



Εικ. 129 Παλέτα Amstrad

Η παλέτα των 27 χρωμάτων σημαίνει ότι, αν και μπορούμε να επιλέξουμε μόνο 16, υπάρχουν πάντα χρώματα για να επιλέξουμε, επιτρέποντάς μας να δημιουργήσουμε σβησίματα και μείγματα. Ωστόσο, άλλοι υπολογιστές της εποχής, όπως ο C64 (ένα σπουδαίο μηχάνημα), είχαν 16 χρώματα από μια παλέτα 16. Ο C64 είχε 3 αποχρώσεις του γκρι μέσα σε μια τόσο μειωμένη παλέτα, κάτι που, αν και έχει επικριθεί, νομίζω ότι δεν είναι κακό, επειδή, καθώς δεν είναι πολύ κορεσμένα χρώματα, συνδυάζονται καλά με το γκρι, και συνδυάζονται επίσης καλά μεταξύ τους, δηλαδή είναι 16 χρώματα που είναι "κοντά" μεταξύ τους.

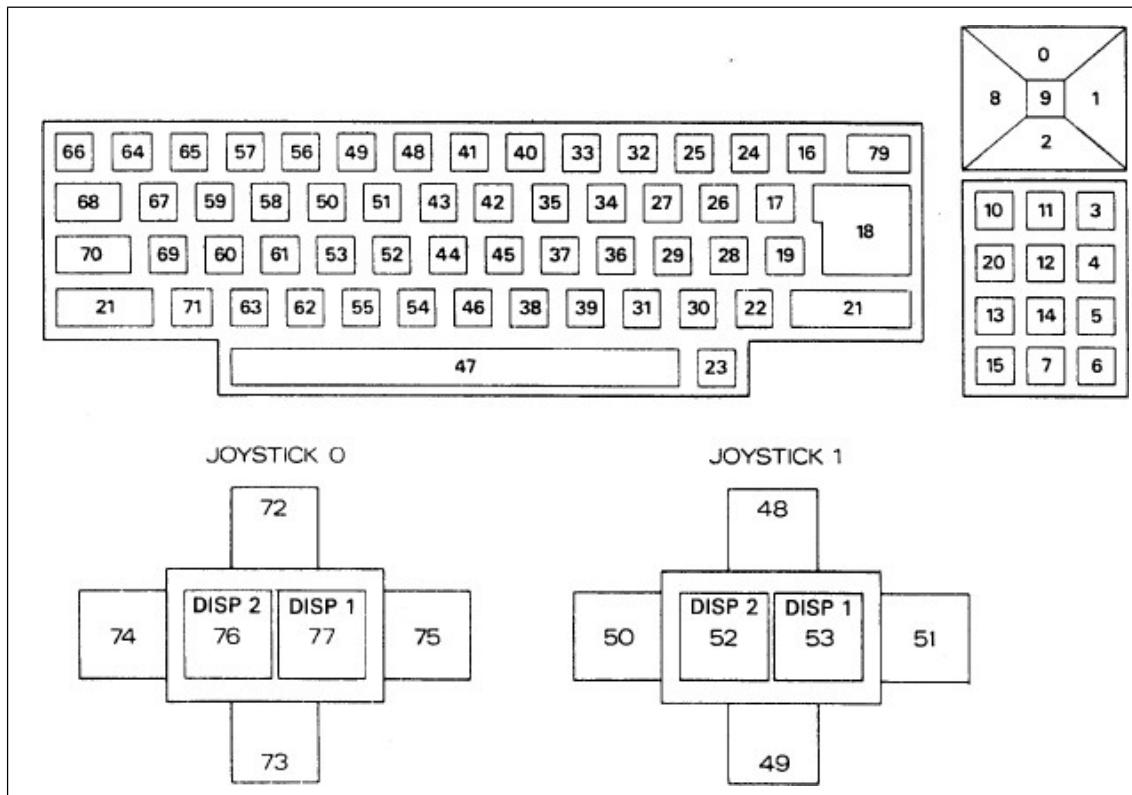


Εικ. 130 Παλέτα C64

Εν ολίγοις, στην Amstrad μπορείτε να επιλέξετε από περισσότερα χρώματα και έτσι να βρίσκετε πάντα το σωστό χρώμα για θόλωση, σκίαση ή απλά για να βρείτε τον χρωματικό τόνο που χρειάζεστε. Η μεγάλη επιτυχία της Amstrad ήταν η δημιουργία μιας παλέτας 27 χρωμάτων, αν και μπορείτε να χρησιμοποιείτε μόνο 16 χρώματα κάθε φορά. Μπορείτε επίσης να επιλέξετε 16 χρώματα που δεν ταιριάζουν καλά μεταξύ τους και να έχετε πολύ κακόγουστα γραφικά (κάτι που είναι αδύνατο στον C64, καθώς δεν μπορείτε να επιλέξετε).

Η παλέτα των 27 επέτρεψε σε πολλά γραφικά φορτίου Amstrad να γίνουν πραγματικά έργα τέχνης.

27 ΠΑΡΑΡΤΗΜΑ III: Κωδικοί INKEY



Κάθε φορά που θέλετε να διαβάσετε το πληκτρολόγιο, προσπαθήστε πρώτα να καθαρίσετε τον απομονωτή ανάγνωσης των τελευταίων πληκτρολογήσεων. Είναι πολύ συχνό το φαινόμενο σε μια οθόνη όπου ζητάτε το όνομα του χρήστη που πέτυχε υψηλό σκορ, να "τρυπώνουν" οι τελευταίες πληκτρολογήσεις του παιχνιδιού (κινήσεις και βολές), εμφανίζοντας πράγματα όπως "OPPPOQQAAA" στην εντολή INPUT. Για να το αποφύγετε αυτό, εκτελέστε κάτι σαν:

**10 B\$=INKEY\$: εάν B\$<>"" ΤΟΤΕ 10
20 INPUT "name:";\$name**

Εκτός από αυτή τη σύσταση, να θυμάστε ότι ο ταχύτερος τρόπος επεξεργασίας του πληκτρολογίου είναι ο εξής:

**10 AN INKEY(keycode) THEN 30: REM μεταβαίνει στο 30 αν δεν πατηθεί
20 <οδηγίες σε περίπτωση που πατηθεί ο
κωδικός πλήκτρου> 30**

29 ΠΑΡΑΡΤΗΜΑ IV: Πίνακας ASCII του AMSTRAD CPC

	0	1	2	3	4	5	6	7	8	9	Α	Β	Γ	Δ	Ε	Φ
0	□	□	□	P	^`	ρ	.	^	α	γ	—	Θ	↑			
1	Γ	Φ	!	1	Α	Q	α	q	▀	β	^	Ι	Θ	↓		
2	Τ	Φ	"	2	Β	R	b	r	▀	„	γ	„	Φ	←		
3	Τ	Φ	#	3	C	S	c	s	▀	£	6	„	†	→		
4	Φ	Φ	*	4	D	T	d	t	.	®,	ε	^	‡	♥	▲	
5	Φ	Φ	X	5	E	U	e	u	▀	Π	Θ)	▼	♣	▼	
6	✓	Π	&	6	F	V	f	v	▀	Γ	λ	▼	○	▶		
7	Ω	Γ	'	7	G	W	g	w	▀	ρ	◀	◀	●	◀		
8	◀	Χ	(8	H	X	h	x	▀	—	14	π	✓	❀	□	
9	→	†)	9	I	Y	i	y	▀	12	σ	ς	▀	▀	▀	
Α	↓	Ω	*	:	J	Z	j	z	▀	—	34	δ	○	❀	δ	
Β	↑	Θ	+	;	K	[k]	▀	±	χ	▀	♀	♂	♂	
Γ	Ψ	Φ	,	<	L	\	l	l	▀	÷	χ	/	▀	J	▀	
Δ	◀	Ω	-	=	M]	m]	▀	—	ω	\	▀	▀	▀	
Ε	Ω	Ω	.	>	N	†	n	~	▀	τ	δ	Σ	▀	▀	▀	
Φ	Θ	Φ	/	?	O	_	o	▀	+	i	Ω	▀	λ	λ	λ	

	0	1	2	3	4	5	6	7	8	9	Α	Β	Γ	Δ	Ε	Φ
0	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
1	1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241
2	2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242
3	3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243
4	4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244
5	5	21	37	53	69	85	101	117	133	149	165	181	197	213	229	245
6	6	22	38	54	70	86	102	118	134	150	166	182	198	214	230	246
7	7	23	39	55	71	87	103	119	135	151	167	183	199	215	231	247
8	8	24	40	56	72	88	104	120	136	152	168	184	200	216	232	248
9	9	25	41	57	73	89	105	121	137	153	169	185	201	217	233	249
Α	10	26	42	58	74	90	106	122	138	154	170	186	202	218	234	250
Β	11	27	43	59	75	91	107	123	139	155	171	187	203	219	235	251
Γ	12	28	44	60	76	92	108	124	140	156	172	188	204	220	236	252
Δ	13	29	45	61	77	93	109	125	141	157	173	189	205	221	237	253
Ε	14	30	46	62	78	94	110	126	142	158	174	190	206	222	238	254
Φ	15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255

30 ΠΑΡΑΡΤΗΜΑ V: ορισμένα ηχητικά εφέ

Πρώτα απ' όλα, πρέπει να ξέρετε ότι το Amstrad έχει 3 κανάλια και τα αναγνωριστικά τους είναι 1, 2 και 4. Για να κάνετε δύο κανάλια ή και τα 3 ταυτόχρονα να ακούγονται, πρέπει απλώς να τα προσθέσετε μαζί.

Χρήση της εντολής SOUND:

SOUND κανάλι, βήμα, διάρκεια, ένταση, βήμα περιβάλλουσας, ένταση περιβάλλουσας, Θόρυβος

ΣΗΜΑΝΤΙΚΟ: Η ένταση κυμαίνεται από 0 έως 7 στο CPC464 και από 0 έως 15 στο 6128. Στο CPC464 μπορούν να χρησιμοποιηθούν οι τιμές 8..15, αλλά αποτελούν επανάληψη των τιμών 0..7 (π.χ. 8 σημαίνει ένταση 0). Αυτή είναι μια ΒΑΣΙΚΗ διαφορά μεταξύ των δύο μοντέλων. Κατά συνέπεια, μια ένταση 10 στο CPC6128 είναι υψηλή ενώ στο 464 είναι πολύ χαμηλή.

Η παράμετρος θορύβου κυμαίνεται από 0 έως 31

Παραδείγματα:

SOUND 1,2000,10,7 : ήχοι καναλιού 1

SOUND 1+2,2000,10,7 : ακούγονται τα κανάλια 1 και 2

Ακολουθούν μερικά παραδείγματα για να τα χρησιμοποιήσετε απευθείας στα προγράμματά σας ή για να σας εμπνεύσουν να δημιουργήσετε άλλους ήχους.

Συλλέξτε ένα διαμάντι ή ένα νόμισμα ENT 1,10,-100,3: ήχος 1,638,30,30,15,15,0,1	χτυπηθήκατε από πέτρα ή βλήμα ENT 1.10, 100.3: ήχος 1,638,30,30,15,15,0,1
Boing ENV 1,1,15,1,15,-1,1: ΉΧΟΣ 1,638,0,0,0,0,1	Boing 2 ENT 2,20,-125,1: ΉΧΟΣ 1,1500,10,12,12,,2
Θάνατος ENT 3,100,5,3: ΉΧΟΣ 1,142,100,15,0,3	
Έκρηξη ΉΧΟΣ 7,1000,20,20,15,,,15	Έκρηξη 2 ENV 1,11,-1,25:ENT 1,9,49,5,9,-10,15 ΉΧΟΣ 3,145,300,12,1,1,1,1,12
Πυροδότηση ENT -5,7,10,1,1,1,7,-10,1: ΉΧΟΣ 1,25,20,12,12,,5	

31 ΠΑΡΑΡΤΗΜΑ VI: Ενδιαφέρουσες ρουτίνες firmware

Σε αυτή την ενότητα θα συμπεριλάβω μερικές ρουτίνες firmware που μπορούν να κληθούν από τη BASIC και που μπορεί να είναι ενδιαφέρουσες στα προγράμματά σας.

CALL 0 : επαναφέρει τον υπολογιστή

CALL &bc02 : αρχικοποιεί την παλέτα στην προεπιλεγμένη τιμή της. Είναι καλή πρακτική να την καλείτε στην αρχή του προγράμματός σας σε περίπτωση που αλλάξει.

CALL &bd19 : συγχρονισμός με τη σάρωση της οθόνης. Αν χειρίζεστε πολύ λίγα sprites μπορείτε να έχετε μια πιο ομαλή κίνηση, αλλά να έχετε υπόψη σας ότι αυτή η εντολή θα επιβραδύνει πολύ το πρόγραμμά σας.

CALL &bb48 : απενεργοποιεί το μηχανισμό BREAK, εμποδίζοντας το πρόγραμμα να σταματήσει εάν εκτελείται.

CALL &bd21 , &bd22, &bd23, &bd24, &bd25 : παράγει ένα εφέ λάμψης στην οθόνη.

Για να επαναφέρετε το ΧΡΟΝΟΔΙΑΚΟΠΗ του AMSTRAD:

Σε ένα 6128

POKE &b8b4,0: POKE &b8b5,0: POKE &b8b6,0: POKE &b8b7,0

Σε ένα 464

POKE &b187,0: POKE &b188,0: POKE &b189,0: POKE &b18a,0

Για να διακρίνετε σε ποιο μηχάνημα βρίσκεται το πρόγραμμά σας, πρέπει να απενεργοποιήσετε τη μουσική και να συμβουλευτείτε μια διεύθυνση με το PEEK.

| MUSIC: Av peek(&39)=57 τότε model=464 αλλιώς model=6128

Av model=464 τότε ...

CALL &bca7 : σταματήστε το κουδούνισμα οποιουδήποτε ήχου που έπαιζε

Η εντολή GRAPHICS PAPER υπάρχει στο CPC6128 αλλά όχι στο CPC464. Ωστόσο, υπάρχει τρόπος να την έχουμε, χρησιμοποιώντας την **CALL &BBE4** και τόσες παραμέτρους με τιμή 1 όσο το χρώμα μελάνης που θέλουμε, για παράδειγμα:

CALL &BBE4,1,1: ίδιο με το "GRAPHICS PAPER 2" αλλά λειτουργεί σε cpc464

CALL &BB18 : περιμένει να πατήσετε ένα πλήκτρο

32 ΠΑΡΑΡΤΗΜΑ VII: Πίνακας χαρακτηριστικών Sprite

Ο παρακάτω πίνακας περιέχει τις διευθύνσεις μνήμης όπου αποθηκεύονται τα χαρακτηριστικά κάθε sprite, καθώς και το μήκος σε bytes του καθενός.

	1byte	2 bytes	2 bytes	1byte	1byte	1byte	1byte	2 bytes	1byte
sprite	status	coordy	coordx	vy	vx	seq	frame	imagen	ruta
0	27000	27001	27003	27005	27006	27007	27008	27009	27015
1	27016	27017	27019	27021	27022	27023	27024	27025	27031
2	27032	27033	27035	27037	27038	27039	27040	27041	27047
3	27048	27049	27051	27053	27054	27055	27056	27057	27063
4	27064	27065	27067	27069	27070	27071	27072	27073	27079
5	27080	27081	27083	27085	27086	27087	27088	27089	27095
6	27096	27097	27099	27101	27102	27103	27104	27105	27111
7	27112	27113	27115	27117	27118	27119	27120	27121	27127
8	27128	27129	27131	27133	27134	27135	27136	27137	27143
9	27144	27145	27147	27149	27150	27151	27152	27153	27159
10	27160	27161	27163	27165	27166	27167	27168	27169	27175
11	27176	27177	27179	27181	27182	27183	27184	27185	27191
12	27192	27193	27195	27197	27198	27199	27200	27201	27207
13	27208	27209	27211	27213	27214	27215	27216	27217	27223
14	27224	27225	27227	27229	27230	27231	27232	27233	27239
15	27240	27241	27243	27245	27246	27247	27248	27249	27255
16	27256	27257	27259	27261	27262	27263	27264	27265	27271
17	27272	27273	27275	27277	27278	27279	27280	27281	27287
18	27288	27289	27291	27293	27294	27295	27296	27297	27303
19	27304	27305	27307	27309	27310	27311	27312	27313	27319
20	27320	27321	27323	27325	27326	27327	27328	27329	27335
21	27336	27337	27339	27341	27342	27343	27344	27345	27351
22	27352	27353	27355	27357	27358	27359	27360	27361	27367
23	27368	27369	27371	27373	27374	27375	27376	27377	27383
24	27384	27385	27387	27389	27390	27391	27392	27393	27399
25	27400	27401	27403	27405	27406	27407	27408	27409	27415
26	27416	27417	27419	27421	27422	27423	27424	27425	27431
27	27432	27433	27435	27437	27438	27439	27440	27441	27447
28	27448	27449	27451	27453	27454	27455	27456	27457	27463
29	27464	27465	27467	27469	27470	27471	27472	27473	27479
30	27480	27481	27483	27485	27486	27487	27488	27489	27495
31	27496	27497	27499	27501	27502	27503	27504	27505	27511

Πίνακας 7 Διευθύνσεις χαρακτηριστικών Sprite

7 ROUTEALL lo ruta	6 Sobre- escritura	5 COLSPALL collider	4 MOVERALL lo mueve	3 AUTOALL lo mueve	2 ANIMALL lo anima	1 COLSP collided	0 PRINTSPALL lo imprime
--------------------------	--------------------------	---------------------------	---------------------------	--------------------------	--------------------------	------------------------	-------------------------------

Πίνακας 8 Σημαίες στο byte κατάστασης

33 ΠΑΡΑΡΤΗΜΑ VIII: Χάρτης μνήμης του 8ΒΡ

ΧΑΡΤΗΣ ΜΝΗΜΗΣ AMSTRAD CPC464 της 8ΒΡ

```
; &FFFF +-----  
; | οθόνη + 8 κρυφά τμήματα των 48bytes το καθένα  
; &C000 +-----  
; | σύστημα (επαναπροσδιορίσιμα σύμβολα, δείκτης  
στοίβας, κ.λπ.)-----  
; 42619 +-----  
; | -Τράπεζα 40 αστέρων (από 42540 έως 42619 = 80bytes)  
; 42540 +-----  
; | Χάρτης διάταξης χαρακτήρων (25x20 =500 bytes)  
; | και παγκόσμιος χάρτης (έως 82 στοιχεία χωράνε σε 500 bytes)  
; | Και τα δύο αποθηκεύονται στην ίδια περιοχή μνήμης.  
; | -γιατί είτε χρησιμοποιείτε το ένα είτε χρησιμοποιείτε το άλλο.  
; 42040 +-----  
; | sprites (σχεδόν 8,5KB για σχέδια).  
; | έχετε 8440 bytes αν δεν υπάρχουν ακολουθίες και  
; | διαδρομές  
; | +-----Οι εικόνες του αλφαριθμητικού αποθηκεύονται επίσης εδώ.  
; | Ορισμοί διαδρομών (μεταβλητού μήκους ο καθένας)  
; | +-----  
; | ακολουθίες κινούμενων σχεδίων των 8 καρέ (16 bytes το  
καθένα)  
; 33600 +-----κειτ-θητέδες ακολουθιών κινούμενων σχεδίων (μακρο-  
; φιλοδοχούμενη)  
; | (1500 Bytes για μουσική που έχει επεξεργαστεί με το  
; 32100 +-----WYZtracker 2.0.1.0)  
; | Ρουτίνες 8ΒΡ (8100 bytes ή 7100 bytes)  
; | Εδώ είναι όλες οι ρουτίνες και ο πίνακας sprite  
; | περιλαμβάνει music player "wyz" 2.0.1.0  
; 25000 +-----  
; |  
; | Η ΛΙΣΤΑ BASIC ή C ΣΑΣ  
; | 24KB, 24,8 KB ή έως και 25KB ελεύθερα για BASIC ή C,  
; | ανάλογα με την επιλογή συναρμολόγησης που  
; | χρησιμοποιείτε για την 8ΒΡ  
; |  
; 0 +-----
```


34 ΠΑΡΑΡΤΗΜΑ ΙΧ: Διαθέσιμες εντολές 8ΒΡ

Κατάλογος των διαθέσιμων εντολών με αλφαριθμητική σειρά:

3D, <flag>, #, offsety 3D, 0	Ενεργοποιεί τη λειτουργία ψευδοτρισδιάστατης προβολής.
ANIMA, #	Αλλάζει το πλαίσιο ενός sprite σύμφωνα με την ακολουθία του
ANIMALL	Αλλάζει το καρέ των sprites με ενεργοποιημένη τη σημαία animation (δεν χρειάζεται να την καλέσετε, αρκεί μια σημαία στην εντολή PRINTSPALL για να την καλέσετε).
AUTO, #	Αυτόματη μετακίνηση ενός sprite σύμφωνα με τα Vy,Vx του
AUTOALL, <flag routed>, <flag routed>, <flag routed>, <flag routed>, <flag routed>.	Κίνηση όλων των sprites με ενεργή τη σημαία αυτόματης κίνησης
COLAY, threshold_ascii, @collision, # COLAY, @collision, # COLAY, # COLAY	Ανιχνεύει τη σύγκρουση με τη διάταξη και επιστρέφει 1 εάν υπάρχει σύγκρουση. Δέχεται μεταβλητό αριθμό παραμέτρων (πάντα με την ίδια σειρά) από 4 έως καμία.
COLSP, #, @collided%, @collided%, @COLSP, #, @collided%, @collided%. COLSP, 32, ini, τέλος COLSP, 33, @collided% COLSP, 34, dy, dx COLSP, #	Επιστρέφει το πρώτο sprite με το οποίο συγκρούεται το#. Η εντολή μπορεί να ρυθμιστεί με τους κωδικούς 32, 33 και 34.
COLSPALL, @who%, @who%, @withwho%. COLSPALL, συγκρουστήρας COLSPALL	Επιστρέφει ποιος συγκρούστηκε (collider) και με ποιον συγκρούστηκε (collided).
LAYOUT, y, x, @string\$, @string\$, @string\$, @string\$, @string\$, @string\$.	Εκτυπώνει λωρίδα εικόνας 8x8 και γεμίζει τη διάταξη χάρτη
LOCATESP, #, y, x	Αλλάζει τις συντεταγμένες ενός sprite (χωρίς να το εκτυπώσει)
MAP2SP, y, x MAP2SP, κατάσταση	Δημιουργεί sprites για να ζωγραφίσει τον κόσμο σε παιχνίδια κύλισης. Τα sprites δημιουργούνται με state = status
MOVER, #, dy, dx	σχετική μετακίνηση ενός μεμονωμένου sprite
OVERALL, dy,dx OVERALL	Σχετική κίνηση όλων των sprites με ενεργή τη σημαία σχετικής κίνησης
MUSIC, C, σημαία, τραγούδι, ταχύτητα MUSIC, σημαία, τραγούδι, ταχύτητα MUSIC	Μια μελωδία αρχίζει να παίζει. Το κανάλι C μπορεί να απενεργοποιηθεί για χρήση με εφέ FX, εάν είναι επιθυμητό. Χωρίς παραμέτρους σταματήστε το κουδούνισμα
PEEK, dir, @variable%	Διαβάζει δεδομένα 16bit (μπορεί να είναι αρνητικά)
POKE, dir, value	εισάγετε ένα δεδομένο 16bit (το οποίο μπορεί να είναι αρνητικό)
PRINTAT, σημαία, y, x, @string	Εκτυπώνει μια συμβολοσειρά επαναπροσδιορίσιμων "μίνι-χαρακτήρων".
PRINTSP, #, y, x PRINTSP, # PRINTSP,32, bits	εκτυπώνει ένα μόνο sprite (# είναι ο αριθμός του) ανεξάρτητα από το byte κατάστασης. Εάν έχει καθοριστεί 32, τότε θέτουμε τα bits φόντου
PRINTSPALL, ini, fin, anima, sync PRINTSPALL, ordermode PRINTSPALL	Εκτυπώνει όλα τα sprites με ενεργή σημαία εκτύπωσης. Εάν κληθεί με μία μόνο παράμετρο, ορίζεται η λειτουργία διάταξης.
RINK,tini,color1,color1,color2,...,colorN RINK, άλμα	Περιστρέφει ένα σύνολο μελανιών σύμφωνα με ένα οριζόμενο μοτίβο που αποτελείται από οποιοδήποτε αριθμό μελανιών

ROUTEESP, #, βήματα	Σας αναγκάζει να περάσετε N βήματα της διαδρομής του sprite ταυτόχρονα.
ROUTEALL	Τροποποιήστε την ταχύτητα του sprite με τη σημαία διαδρομής (δεν χρειάζεται να την καλέσετε, απλά σημειώστε τη σημαία στο AUTOALL).
SETLIMITS, xmin, xmax, ymin, ymax	Ορίζει το παράθυρο του παιχνιδιού, όπου γίνεται η αποκοπή.
SETUPSP, #, param_number, value SETUPSP, #, 5, Vy, Vx	Τροποποιεί μια παράμετρο ενός sprite. Εάν έχει καθοριστεί η παράμετρος 5, μπορεί να παρέχεται προαιρετικά η Vx.
STARS, initstar, num, color, dy, dx	Κύλιση ενός συνόλου αστεριών
UMAP,adr_ini, adr_end, yini, yfin, xini, xfin	Ενημερώνει τα στοιχεία του παγκόσμιου χάρτη με ένα υποσύνολο στοιχείων από έναν μεγαλύτερο χάρτη

35 ΠΑΡΑΡΤΗΜΑ Χ: Επιλογές συναρμολόγησης 8BP

Από την έκδοση V42, η βιβλιοθήκη 8BP διαθέτει διάφορες επιλογές συναρμολόγησης που σας επιτρέπουν να επιλέξετε τις χωρητικότητες που θέλετε για το παιχνίδι σας και έτσι να έχετε περισσότερη μνήμη διαθέσιμη για την καταχώριση του παιχνιδιού σας.

Η επιλογή συναρμολόγησης πρέπει να καθοριστεί στο αρχείο **Make_all_mygame.asm**, το οποίο έχει μια συγκεκριμένη γραμμή για να εκχωρήσει την τιμή της παραμέτρου "**ASSEMBLING_OPTION**".

Επιλογή	Περιγραφή της επιλογής	Παράδειγμα τυπικού παιχνιδιού
0	<p>Μπορείτε να κάνετε οποιοδήποτε παιχνίδι Όλες οι διαθέσιμες εντολές Πρέπει να χρησιμοποιήσετε το MEMORY 23499</p> <p>Για να αποθηκεύσετε τη βιβλιοθήκη + γραφικά + μουσική: SAVE "8BP0.bin",b,23500,19119</p>	οποιοσδήποτε
1	<p>παιχνίδια λαβύρινθου ή διέλευσης οθόνης Πρέπει να χρησιμοποιήσετε τη μνήμη 24999 Δεν διατίθεται σε αυτή τη λειτουργία:</p> <p> MAP2SP, UMAP, 3D</p> <p>Για να αποθηκεύσετε τη βιβλιοθήκη + γραφικά + μουσική: ΑΠΟΘΗΚΕΥΣΗ "8BP1.bin",b,25000,17619</p>	
	<p>Για παιχνίδια κύλισης Πρέπει να χρησιμοποιήσετε το MEMORY 24799 Δεν είναι διαθέσιμο σε αυτή τη λειτουργία:</p> <p> LAYOUT, COLAY, 3D</p> <p>Για να αποθηκεύσετε τη βιβλιοθήκη + γραφικά + μουσική: ΑΠΟΘΗΚΕΥΣΗ "8BP2.bin", b,24800,17819</p>	
	<p>Για παιχνίδια με ψευδο 3D Πρέπει να χρησιμοποιήσετε το MEMORY 23999 Δεν είναι διαθέσιμο σε αυτή τη λειτουργία:</p> <p> LAYOUT, COLAY</p> <p>Για να αποθηκεύσετε τη βιβλιοθήκη + γραφικά + μουσική: ΑΠΟΘΗΚΕΥΣΗ "8BP3.bin",</p>	

	b,24000,18619	
--	----------------------	--

36 ΠΑΡΑΡΤΗΜΑ XI: Αντιστοίχιση RSX/CALL

Κατάλογος των διαθέσιμων εντολών με αλφαριθμητική σειρά και η σχετική τους διεύθυνση για τη χρήση της κλήσης CALL &XXXX όταν είναι απαραίτητο για την αύξηση της ταχύτητας. Έχω δώσει μόνο μερικά ενδεικτικά παραδείγματα, καθώς η χρήση είναι πανομοιότυπη με την εντολή RSX, με τη διαφορά ότι πρέπει να αντικαταστήσετε την εντολή με CALL <διεύθυνση>.

ΣΗΜΑΝΤΙΚΟ: Από τη μία έκδοση της 8BP στην άλλη, αυτός ο κατάλογος διευθύνσεων μπορεί να διαφέρει. Βεβαιωθείτε ότι χρησιμοποιείτε την τελευταία έκδοση της 8BP, αν πρόκειται να χρησιμοποιήσετε τις διευθύνσεις αυτού του πίνακα.

COMMAND	ΔΙΕΥΘΥΝΣΗ	ΠΑΡΑΔΕΙΓΜΑ
3D	&6BDE	
ANIMA	&6BB7	
ANIMALL	&7479	
AUTO	&6BC9	
AUTOALL	&6B9C	CALL &6B9C,1
COLAY	&6BA8	
COLSP	&6BBA	
COLSPALL	&6B99	
LAYOUT	&6BD5	
LOCATESP	&6BAE	
MAP2SP	&6BA2	
MOVER	&6BC0	
MOVERALL	&6B9F	
MUSIC	&6BD8	ΚΛΗΣΗ &6BD8,0,0,0,0,0,0,6
PEEK	&6BB1	CALL &6BB1,dir,@var
POKE	&6BB4	
PRINTAT	&6BC6	CALL &6BC6,0,y,x,@c\$ CALL &6BC6,0,y,x,@c\$
PRINTSP	&6BC3	CALL &6BC3,31
PRINTSPALL	&6B96	ΚΛΗΣΗ &62A6,0,0,0,0,0,0,0,0
RINK	&6BBD	
ROUTESP	&6BCC	
ROUTEALL	&6BD2	
SETLIMITS	&6BDB	
SETUPSP	&6BAB	
STARS	&6BA5	
UMAP	&6BCF	

37 ΠΑΡΑΡΤΗΜΑ XII: ΛΕΙΤΟΥΡΓΙΕΣ της 8BP σε C

RSX	Πρωτότυπο C
3D, 0 3D, <flag>, #, offsety	void _8BP_3D_1(int flag), void _8BP_3D_3(int flag, int sp_fin, int offsety),
ANIMA, #	void _8BP_anima_1(int sp),
ANIMALL	void _8BP_animall(),
AUTO, #	void _8BP_auto_1(int sp),
AUTOALL, <flag routed>, <flag routed>, <flag routed>, <flag routed>, <flag routed>.	void _8BP_autoall(), void _8BP_autoall_1(int flag),
COLAY, threshold_ascii, @collision, # COLAY, @collision, # COLAY, # COLAY	void _8BP_colay_3(int threshold, int* collision, int sp), void _8BP_colay_2(int* collision, int sp), void _8BP_colay_1(int sp); void _8BP_colay(),
COLSP, #, @collided%, @collided%, @COLSP, #, @collided%, @collided%. COLSP, 32, ini, τέλος COLSP, 33, @collided% COLSP, # COLSP, 34, dy, dx	/* λειτουργία 32, ini,fin ή λειτουργία 34,dy,dx*/ void _8BP_colsp_3(int operation, int a, int b), /*λειτουργία 33 ή sp*/ void _8BP_colsp_2(int sp, int* collision); void _8BP_colsp_1(int sp),
COLSPALL,@who%,@who%,@with whom% COLSPALL, συγκρουστήρας COLSPALL	void _8BP_colspall_2(int* collider, int* collided); void _8BP_colspall_1(int collider_ini), void _8BP_colspall(),
LAYOUT, y, x, @string\$, @string\$, @string\$, @string\$, @string\$, @string\$, @string\$, @string\$.	void _8BP_layout_3(int y, int x, char* cad),
LOCATESP, #, y, x	void _8BP_locatesp_3(char sp, int y, int x),
MAP2SP, y, x MAP2SP, κατάσταση	void _8BP_map2sp_2(int y, int x), void _8BP_map2sp_1(unsigned char status),
MOVER, #, dy, dx	void _8BP_mover_3(int sp, int dy,int dx); void _8BP_mover_1(int sp),
MOVERALL, dy,dx	void _8BP_moverall_2(int dy, int dx); void _8BP_moverall(),
MUSIC, C, σημαία, τραγούδι, ταχύτητα MUSIC, σημαία, τραγούδι, ταχύτητα MUSIC	void _8BP_music_4(int flag_c, int flag_repetition,int song, int speed), void _8BP_music(),
PEEK, dir, @variable%	void _8BP.Peek_2(int address, int* data),
POKE, dir, value	void _8BP.poke_2(int address, int data),
PRINTAT, σημαία, y, x, @string	void _8BP.printat_4(int flag,int y,int x,char* cad),
PRINTSP, #, y, x PRINTSP, # PRINTSP,32, bits	void _8BP.printsp_1(int sp) , void _8BP.printsp_2(int sp, int bits_background) ; void _8BP.printsp_3(int sp,int y,int x) ,
PRINTSPALL, ini, fin, anima, sync PRINTSPALL, ordermode PRINTSPALL	void _8BP.printspall_4(int ini, int fin, int flag_anima, int flag_sync), void _8BP.printspall_1(int order_type); void _8BP.printspall(),
RINK,tini,color1,color1,color2,...,color N RINK, άλμα	void _8BP_rink_N(int num_params,int* ink_list); void _8BP_rink_1(int step),
ROUTESP, #, βήματα	void _8BP_routesp_2(int sp, int steps)- void _8BP_routesp_1(int sp),
ROUTEALL	void _8BP_routeall(),
SETLIMITS, xmin, xmax, ymin, ymax	Void _8BP_setlimits_4(int xmin, int xmax, int ymin, int ymax)

SETUPSP, #, param_number, value	void _8BP_setupsp_3(int sp, int param, int value)- void _8BP_setupsp_4(int sp, int param, int value1,int value2),
STARS, initstar, num, color, dy, dx	void _8BP_stars_5(int star_ini, int num_stars,int color, int dy, int dx), void _8BP_stars(),
UMAP,adr_ini, adr_end, yini, yfin, xini, xfin	void _8BP_umap_6(int map_ini, int map_fin, int y_ini, int y_fin, int x_ini, int x_fin),

38 ΠΑΡΑΡΤΗΜΑ XIII: MiniBASIC σε C

BASIC	Πρωτότυπο C
BORDER	<code>void _basic_border(char color), //παράδειγμα _basic_border(7)</code>
ΚΑΛΕΣΤΕ	<code>void _basic_call(unsigned int address), // παράδειγμα _basic_call(0xbd19)</code>
DRAW	<code>void _basic_draw(int x, int y),</code>
INK	<code>void _basic_ink(char ink1,char ink2),</code>
INKEY	<code>char _basic_inkey(char key), //διαρκεί περίπου 0,3 ms. αργό αλλά απλό</code>
ΤΟΠΟΘΕΤΗΣ Η	<code>void _basic_locate(unsigned int x, unsigned int y), // παράδειγμα: _basic_locate(2,25);_basic_print("TEST"),</code>
MOVE	<code>void _basic_move(int x, int y),</code>
XAPTI	<code>void _basic_paper(char ink),</code>
PEEK	<code>char _basic.Peek(unsigned int address),</code>
GRAPHICS PEN	<code>void _basic_pen_graph(char ink),</code>
PEN	<code>void _basic_pen_txt(char ink),</code>
POKE	<code>void _basic_poke(unsigned int address, unsigned char δεδομένα),</code>
PLOT	<code>void _basic_plot(int x, int y),</code>
ΕΚΤΥΠΩΣΗ	<code>void _basic_print(char *cad), //παράδειγμα: _basic_print("Hello")</code>
RND	<code>unsigned int _basic_rnd(int max), //παράδειγμα: num=_basic_rnd(50)</code>
ΗΧΟΣ	<code>void _basic_sound(unsigned char nChannelStatus, int nTonePeriod, int nDuration, unsigned char nVolume, char nVolumeEnvelope, char nToneEnvelope, unsigned char nNoisePeriod),</code>
STR\$	<code>char* _basic_str(int num), //όμοια με το STR\$ //παράδειγμα: _basic_print(_basic_str(num))</code>
ΧΡΟΝΟΣ	<code>unsigned int _basic_time(), //επιστρέφει ένα unsigned int,(0..65535). Ως ακέραιος, όταν // φτάνουμε στο 32768 πηγαίνουμε στο -32768</code>