P.S.I.

MICRO ET TECHNIQUES

CRÉATION ET ANIMATIONS GRAPHIQUES SUR AMSTRAD CPC

GILLES FOUCHARD
ET JEAN-YVES CORRE

CRÉATION ET ANIMATIONS GRAPHIQUES SUR AMSTRAD CPC

Connaissez-vous toute la collection Amstrad chez P.S.I. ?

Pour les Amstrad CPC 464, 664 et 6128 :

Initiation

— La découverte de l'Amstrad — Daniel-Jean David
— Exercices en Basic pour Amstrad — Maurice Charbit

Programmation BASIC

— 102 programmes pour Amstrad — Jacques Deconchat
— Super jeux pour Amstrad — Jean-François Sehan
— Amstrad en famille — Jean-François Sehan
— Super générateur de caractères sur Amstrad — Jean-François Sehan
— Photographie sur Amstrad et Apple II — Pierrick Moigneau et Xavier de la Tullaye
— Amstrad en musique — Daniel Lemahieu

Maîtrise du BASIC

— Basic Amstrad, 1. Méthodes pratiques — Jacques Boisgontier et Bruno Césard
— Basic Amstrad, 2. Programmes et fichiers — Jacques Boisgontier
— Basic Plus, 80 routines sur Amstrad — Michel Martin
— Périphériques et fichiers sur Amstrad — Daniel-Jean David

Langages

— Assembleur de l'Amstrad — Marcel Henrot
— Graphisme en assembleur sur Amstrad CPC — Francis Piérot
— Trois étapes vers l'intelligence artificielle sur Amstrad CPC — René Descamps
— Turbo Pascal sur Amstrad — Pierre Brandeis et Frédéric Blanc
— Clefs pour dBASE II et III — Michel Keller

Système

— Clefs pour Amstrad, 1. Système de base — Daniel Martin
— CP/M Plus sur Amstrad 6128 et 8256 — Yvon Dargery
— Clefs pour Amstrad, 2. Système disque — Daniel Martin et Philippe Jadoul

A paraître :

— Intelligence artificielle : langage et formes sur Amstrad — Thierry Lévy-Abégnolli et Olivier Magnan
— Clefs pour Amstrad 8256 — Eric Baumarti

Pour tout problème rencontré dans les ouvrages P.S.I. vous pouvez nous contacter au numéro ci-dessous :

Numéro Vert/Appel Gratuit en France

05 21 22 01

(Composer tous les chiffres, même en région parisienne)

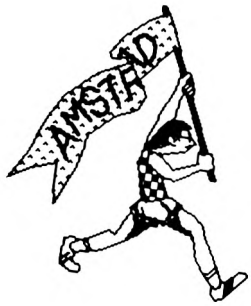
ISBN : 2-86595-338-6

MICRO ET TECHNIQUES

CRÉATION ET ANIMATIONS GRAPHIQUES SUR AMSTRAD CPC

GILLES FOUCHARD
ET JEAN-YVES CORRE

ÉDITIONS DU P.S.I.
1986



Sommaire

Présentation

Ce livre s'adresse à toutes les personnes désireuses de développer conjointement leurs connaissances techniques et leur sens créatif.

Nous voulons donner le goût de la création d'images aux "branchés" de la micro, et convaincre les autres que la micro-informatique est un outil pour la création, au même titre que les outils "classiques" (dessin, photo, vidéo...).

Le premier chapitre traite de la **création graphique,** décrivant les outils (matériels et logiciels) de création sur Amstrad 464, 664 et 6128. Nous proposons des exemples pratiques de dessin interactif à l'aide de la souris ou du joystick (ces programmes fonctionnent sur l'ensemble de la gamme Amstrad).

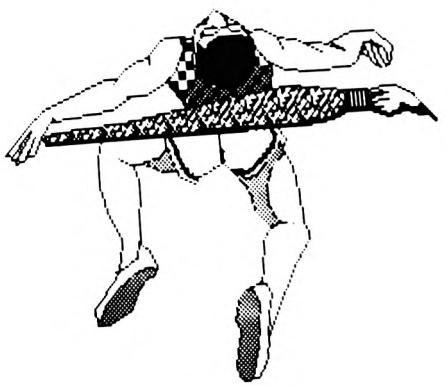
Le deuxième chapitre traite de **l'animation des images ;** ce sujet est également illustré par des exemples de réalisations pratiques (une des animations ne peut fonctionner que sur le modèle 6128, car elle nécessite 128K de mémoire centrale). Vous pourrez – à l'aide des programmes présentés – réaliser vos propres animations, et imaginer d'autres principes d'animation. Alors, si vous avez des images plein la tête, tous à vos claviers...

Ce livre n'est pas un manuel de référence du Basic Amstrad ou de l'assembleur Z-80, mais la diversité des exemples proposés peut constituer un excellent outil d'apprentissage de ces langages. Vous trouverez dans ce document l'ensemble des listes sources Basic et Assembleur ; nous avons préféré décrire chaque programme de manière concise, les explications sur le fonctionnement et les méthodes de programmation choisies étant spécifiées en commentaires à l'intérieur des sources ; cela vous évitera donc des allées et venues pénibles entre un texte explicatif et un programme source.

Vous pouvez d'autre part vous procurer par correspondance la disquette d'accompagnement (face A : Création graphique / face B : Animations) contenant l'ensemble des programmes sources et "exécutables", ainsi que les images couleur présentées en couverture et en illustration interne.

Enfin, en annexe, sont regroupés le guide utilisateur de la disquette et la description des catalogues, la liste des routines utilisées dans les programmes (point d'entrée dans la ROM, et description) ainsi qu'une bibliographie.

CREATION GRAPHIQUE



Introduction

L'utilisation des outils informatiques engendre une nouvelle forme de communication visuelle ; ainsi, le "gestionnaire" manipule des images tout comme Monsieur Jourdain faisait de la prose : un écran est un ensemble de fenêtres permettant de jeter un regard sur des lieux d'information "éloignés" ; telle fenêtre est un regard sur une fiche client par exemple, et telle autre visualise un état des commandes du client ; à une question fonctionnelle est associée une image synthétique. La représentation de la réponse à l'écran n'est pas unique : le choix étant guidé par des considérations techniques (limites de l'écran, organisation des fichiers...) et ergonomiques (facilité d'enchaînement sur une autre tâche : utilisation des menus déroulants, compréhension visuelle, utilisation de symboles ou icônes désignant une action précise...) ; on pourra faire appel à des représentations schématiques : graphes, histogrammes ; enfin, l'utilisation de la couleur sera un atout supplémentaire.

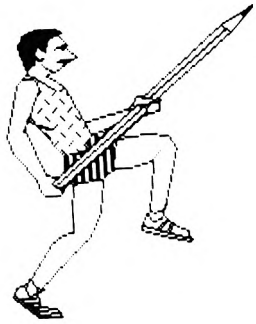
Dans un tout autre domaine, celui des jeux, la production de graphismes de plus en plus sophistiqués est nécessaire pour satisfaire les consommateurs les plus exigeants. Les jeux d'adresse font principalement appel à l'animation de motifs (personnages, objets...) ; les images "plein écran" correspondant aux différents degrés de difficulté du jeu. La qualité graphique des images est souvent meilleure dans les jeux d'aventure (ou de rôle), où le réalisme des décors conditionne le moral du joueur ; on s'investit beaucoup plus lorsque les images vous "parlent", et on franchit alors le pas qui vous conduit dans un monde imaginaire.

Le domaine ludique rejoint le domaine éducatif où l'image devient prépondérante dans les logiciels s'adressant à de très jeunes enfants ; un programme de coloriage est-il un exemple d'apprentissage... du

coloriage d'une image ou de l'utilisation d'un micro-ordinateur ? En fait, l'ordinateur est un outil bien vite oublié par l'enfant, qui ne voit plus que son "crocodile bleu et vert".

Finalement, l'ordinateur est un outil d'expérimentation idéal pour la création graphique ; les possibilités de transformation rapide de l'image (inversion de couleur par exemple) sont séduisantes pour les graphistes ; ainsi, la couverture de ce livre et les illustrations internes ont été fabriquées sur un Amstrad 664 en mode 16 couleurs.

Mais la communication visuelle est d'autant plus attrayante que nous disposons d'outils bien adaptés à notre dialogue avec le micro-ordinateur ; la qualité de la communication exige en effet un comportement plus naturel avec la machine : la reconnaissance vocale en est l'exemple le plus sophistiqué ; nous serons moins exigeant, la "souris" ou le joystick permettent de déplacer des objets à l'écran et de donner des ordres en "cliquant" ; le gestionnaire utilisera la souris pour fabriquer du texte : il "coupe", "copie" et "colle" des morceaux de texte à l'aide de la souris ; l'enfant de quatre ans manipule la souris avec autant d'aisance, il va "cliquer" sur un pinceau représenté à l'écran, choisir une couleur et colorier son crocodile ; nous utiliserons la souris (ou le joystick) pour tracer, gommer, "vaporiser", ou définir et déplacer des fenêtres de travail à l'intérieur desquelles nous pourrons transformer l'image (zoom, symétries et inversion de couleurs).



Chapitre 1

LES OUTILS

LES MATÉRIELS AMSTRAD ET LES PÉRIPHÉRIQUES

Les différents modèles de la gamme Amstrad sont commercialisés avec un moniteur vidéo (monochrome ou couleur) ; si vous désirez connecter l'unité centrale à votre téléviseur, vous devrez vous procurer un interface Péritel couplé à un boîtier d'alimentation spécifique (en effet les moniteurs Amstrad contiennent l'alimentation spécifique de l'unité centrale) ; tous ces écrans de visualisation fonctionnent suivant le même principe : le balayage de trame (une autre famille d'écrans cathodiques utilise la technologie du balayage cavalier : ces écrans graphiques spécialisés sont utilisés principalement en CAO).

Le critère principal d'évaluation d'un écran de visualisation (à balayage de trame) est sa définition : elle correspond au nombre de points affichables par ligne et au nombre de lignes balayées ; mais la définition réelle de l'image sera en dernier ressort conditionnée par les caractéristiques techniques de l'UC (et par le mode de gestion de l'image utilisée : voir à ce sujet le chapitre 2). Il ne faut donc pas s'affoler si un cercle présente des marches d'escalier, c'est plus pratique pour en faire le tour !

On retrouve cette notion de qualité de définition dans les supports traditionnels en dessin (le grain de la feuille) ; l'image agrandie d'un trait de crayon sur une feuille ne donne pas une ligne continue : c'est un tracé discontinu – comme le sont les points/pixels de l'écran – mais

de nature irrégulière. Il n'y a pas de définition idéale de l'image, un seuil en dessous duquel une image n'aurait plus de valeur : tout dépend de l'effet et de la précision recherchée ; cependant, si l'on cherche à représenter une image réaliste d'un objet, la précision recherchée exige la meilleure définition... au coût le plus supportable

Le grain important de certaines émulsions photographiques peut être recherché volontairement ; de même, une faible définition électronique peut apporter un effet visuel désiré ; une contrainte technique (mauvaise définition, flou de hachures...) peut devenir un atout pour une réalisation particulière. Une faible quantité d'informations visualisées peut conduire à une lecture parfaite, l'art résidant dans l'agencement particulier des "informations" disponibles.

Les critères d'évaluation indépendants du micro-ordinateur sont la taille du point et les caractéristiques de contraste, de luminosité et de saturation des couleurs ; mais la "maison" AMSTRAD ne vous laisse pas le choix du moniteur. Les illustrations couleur du livre ont été réalisées à partir de photographies d'écran du moniteur vidéo couleur CTM644.

Vous possédez une "feuille-écran", il vous faut maintenant choisir votre outil de saisie : votre "crayon à tout faire" ; en effet, un outil unique réalisera les fonctions de crayon/gomme/pinceau qui sont prises en charge par le logiciel.

L'outil de saisie traditionnel est le clavier ; aujourd'hui indispensable – pour le développement et la mise au point des logiciels, ou la saisie de texte et de chiffres –, son emploi peu convivial tend à être minimisé dans la création graphique, laissant la place à des outils plus interactifs.

Les outils de fabrication de l'image

- ☐ **Le stylo optique** (light-pen) : il doit permettre de désigner des fonctions ou des objets en les pointant à l'écran ; le logiciel interprète alors le point visé et "réagit" en conséquence ; cependant, le crayon actuellement distribué par Amstrad ne possède pas d'interrupteur, les sélections doivent donc être validées au clavier ; cette conception est regrettable et pénalisante pour une application de dessin.
- ☐ **La manette "balai"** (joystick) est principalement utilisée dans les jeux vidéo ; elle permet de dessiner de façon agréable car votre liberté de mouvement est plus grande et implique plus de spontanéité dans vos tracés. Mais la plupart des modèles manquent de précision pour le dessin ou se dégradent à l'usage, surtout si vous êtes un boulimique des jeux d'adresse.

☐ Enfin, la souris – **AMX MOUSE** – est mieux adaptée au dessin que le joystick, car elle est plus maniable et précise.
Elle a besoin d'une surface plane et lisse de préférence pour évoluer, alors si votre table de travail est un "vrai chantier", il faudra réviser vos méthodes de travail pour laisser un peu de place à votre souris. Ce petit boîtier roulant sur votre table déplace le curseur à l'écran et vous permet de sélectionner une fonction ou un outil de dessin en "cliquant" le bouton du boîtier. AMX MOUSE est livrée avec un logiciel de dessin AMX ART.

Les outils de restitution

L'image affichée sur l'écran de visualisation réside dans la mémoire vive du micro-ordinateur ; cette mémoire étant volatile, vous devez donc stocker vos travaux sur une mémoire annexe – disquette ou cassette – en vue d'un ré-affichage et traitement ultérieur.

Pour restituer vos travaux, vous pouvez utiliser l'imprimante DMP 2000 qui possède six modes graphiques imprimant de 480 à 1 920 points par ligne ; ces modes peuvent être programmés en Basic Amstrad.

Pour conserver une "trace" couleur de vos travaux, la seule solution est la photographie d'écran ; un matériel photographique standard donne entière satisfaction, à vous de trouver la meilleure façon de procéder.

LES LOGICIELS GRAPHIQUES

Un logiciel graphique met à la disposition de l'utilisateur un ensemble de fonctions que nous présentons ci-après.

Les fonctions interactives de dessin

Ces fonctions réalisent des tracés par pilotage d'un outil de saisie : souris, joystick...

☐ **Le tracé d'un point :** le déplacement de la souris positionne un curseur (ou un autre objet symbolique, crayon) à l'écran ; on clique, on "relâche » le bouton et le point est fixé dans la couleur en cours ; cette fonction est marginale : il n'est pas question de construire une image point par point... par contre, cette fonction sera utilisée pour construire un motif (faible taille) ou une trame, ou pour travailler à la "loupe" un détail de l'image...

☐ **Le tracé d'une ligne** : on peut retenir deux méthodes : on clique, on relâche, on déplace, on clique à nouveau, on relâche, la droite est tracée ; l'autre méthode permet de visualiser le tracé avant l'enregistrement définitif en mémoire : on clique, on maintient le bouton de la souris enfoncé, on déplace la souris, on "étire" donc la ligne, on change l'angle de tracé, la "ligne" est tracée dès que l'on relâche le bouton.

☐ **Le tracé libre** permet, tout en maintenant le bouton enclenché, de réaliser l'ébauche spontanée d'un dessin : c'est la fonction de base du dessin ; elle traduit le mouvement de la main ; la réponse au mouvement est tributaire de la sensibilité de la souris, qu'il peut être bon de modifier dans certains cas.

☐ **Le tracé de lignes "continues"** est un prolongement du tracé de ligne ; le départ d'une ligne coïncide avec le point d'arrivée de la ligne précédente.

☐ **Le tracé de "rayons"** : toutes les lignes convergent vers un point unique.

☐ **Le tracé de rectangle.**

☐ **Le tracé de cercle.**

☐ **Le tracé d'ellipse.**

☐ **L'aérographe** : cette fonction consiste à tracer une grille de points espacés ; on pourra réaliser des "ombrages" en "vaporisant" plus ou moins sur une zone de l'image.

☐ **La gomme** : elle efface, en fait elle trace en noir.

Les fonctions interactives évoluées

☐ **Le coloriage** permet de remplir une surface fermée : la surface est délimitée par un contour ; s'il manque un point sur ce contour, la couleur se "déversera" dans d'autres parties de l'image ! Il est donc à utiliser avec précaution lorsque l'on n'a pas la possibilité d'annuler la dernière commande.
La couleur de coloriage peut être une trame.

☐ **Le dessin** à la loupe (zoom) : une partie de l'image est agrandie, un point étant représenté par un rectangle de points ; on "trace" dans la zone agrandie, les modifications sont répercutées sur l'image initiale.

☐ **La fabrication d'objets et de motifs** et l'édition de ces objets sur une image ; un objet correspond à un tracé que l'on peut "copier" en différents endroits de l'image. Une extension de l'utilisation des tables d'objets est le jeu de pinceaux et brosses disponibles ;

chaque brosse étant un objet particulier avec lequel on peut dessiner : le programme réalise un tracé continu de la brosse sélectionnée.

Les fonctions de "mise en page"

☐ **Le découpage** (capture / cut).
☐ **La copie** (copy).
☐ **Le collage** (paste).

Ces fonctions constituent le "noyau" d'un éditeur graphique ; les actions de "mise en page" sont similaires aux fonctions évoluées d'un traitement de texte (cut/copy/paste) ; mais au lieu d'agir sur des mots ou paragraphes, on agit sur une matrice de points délimités par une fenêtre.

Certains logiciels (SuperPaint chez Micro-Application) permettent le détourage d'une forme : la "capture au lasso" permet de découper une forme quelconque ; dans la plupart des cas, la capture est rectangulaire (fenêtre).

Une zone "capturée" peut être déplacée et collée à un autre endroit de l'image ; on peut également recopier la zone dans tout emplacement de la page graphique.

Les fonctions de transformation de l'image par calcul

On peut distinguer deux classes de transformations : les transformations géométriques et les variations sur les couleurs.

☐ **Les transformations géométriques** : les fonctions les plus classiques sont les fonctions "miroirs" qui permettent de renverser l'image (dans le sens vertical ou horizontal) ; l'opération peut être interactive : vous avez choisi une symétrie par rapport à une verticale partageant l'écran en deux zones ; votre tracé dans la partie gauche sera recopié et inversé dans la partie droite en "temps réel".

On peut envisager la réduction d'une image ; compte tenu des résolutions disponibles sur Amstrad, l'algorithme utilisé peut conduire à une image réduite illisible ; il est conseillé d'utiliser la réduction sur des images "dépouillées" (larges tracés en noir et blanc par exemple).

☐ **Les variations sur les couleurs** : trois modes sont intéressants pour le graphiste et conduisent à d'étonnantes variations sur une même image :
- l'inversion des couleurs ;
- l'échange de deux couleurs ;
- le remplacement d'une couleur par une autre.

Un atout supplémentaire est de pouvoir activer ces fonctions sur une partie "capturée" de l'image.

La magie de l'ordinateur réside dans ces possibilités de jouer avec les couleurs ; votre travail sera démultiplié, vous pouvez ainsi, par essais successifs, rechercher le meilleur effet visuel.

Les fonctions de définition de l'environnement de travail

Ces fonctions déterminent le choix de vos outils ; pour dessiner et "peindre", vous devez sélectionner vos tubes et vos pinceaux, vous devez préparer vos mélanges de couleurs...

☐ **La gestion de la palette de couleurs** : en fonction du mode graphique utilisé, cette fonction vous permet de choisir une couleur de travail en tracé libre ou en mode "coloriage"...

☐ **La fabrication de trames** : c'est une fonction essentielle lorsque l'on dispose d'une palette réduite (4 ou 16 couleurs) car la trame de points crée une impression de couleur supplémentaire ; d'autre part, la trame peut donner une impression de matière (texture) particulière.

Cette fonction est essentielle certes, mais attention, même en mode 16 couleurs, on reste limité : à vous d'exploiter au mieux les contraintes techniques.

Les trames peuvent être pré-définies ou construites selon les désirs du dessinateur ; on dispose dans ce cas d'une grille (matrice de points) de travail dans laquelle on dispose des points de couleurs ; la répétition de la trame ainsi construite permettra de colorier une surface ou de "peindre" cette trame avec une brosse large.

Plus la palette de couleurs est grande, plus il est intéressant d'utiliser une large grille de fabrication afin d'augmenter les possibilités d'effets tramés ; ainsi, une grille de 8 par 8 sera idéale ; une grille plus large présenterait cependant peu d'intérêt, compte tenu de la résolution, une grille de 4 par 4 serait par contre insuffisante.

☐ **Le jeu de pinceaux ou brosses** : un logiciel graphique propose un jeu de brosses très utile en mode "peinture" ; une brosse est une forme géométrique qui conditionne votre tracé.

Les fonctions texte

Une image peut comporter du texte ; vous avez le choix entre plusieurs polices de caractères, chaque lettre est en fait une forme particulière ; il peut être intéressant de fabriquer ses propres caractères : cependant, la définition de l'image ne vous permettra pas de reproduire les styles du catalogue Letraset.

Les utilitaires

Il s'agit des fonctions d'entrée-sortie, c'est-à-dire de stockage (sauvegarde) ou de rappel (chargement en mémoire centrale) d'une image, mais aussi d'une fenêtre d'image (window), d'une forme ou d'une table de formes, d'un jeu de brosses, d'une police de caractères, ou d'une palette de trames.

A cet ensemble de fonctions, on peut ajouter :

- le catalogue d'une disquette ;
- le formatage d'une disquette de travail ;
- la suppression de fichiers "image".

Les fonctions diverses

Une fonction non négligeable est la fonction d'annulation de la dernière commande. Cette fonction est vitale en mode "bit-map" où l'image se réduit à un ensemble de points et ne garde donc pas de trace des actions de dessin ; un logiciel peut fonctionner en mode "vecteur" ; c'est le cas des logiciels orientés "3D" : le fichier image est un ensemble d'actions (commande et paramètres de la commande) ; ainsi, on enregistre la commande "tracé d'une droite" et ses paramètres (point de départ et d'arrivée), au lieu de mémoriser l'ensemble des points constituant la droite ; il est donc possible de modifier une commande quelconque de la fabrication de l'image ; le fichier image en mode "vecteur" correspond à une liste d'affichage ; lors du rappel d'une telle image, on traduit les commandes et on visualise donc l'historique du tracé initial.

Enfin, un logiciel peut être auto-documenté, une fonction d'aide (help) étant disponible à tout instant de la fabrication de l'image. La standardisation et le symbolisme visuel des fonctions (menus déroulants, fenêtres de travail, icônes...) réduit le besoin de documentation.

Nous citons ci-après les principales familles de logiciels ou utilitaires graphiques :

- les logiciels de dessin comme SuperPaint de Micro-Application ;
- les utilitaires de représentation d'objets en trois dimensions ;
- les utilitaires de recopie d'écran sur une imprimante ;

- les logiciels d'aide à la programmation tel que HBASIC de Power-Soft, qui est une extension du Basic de l'Amstrad offrant 46 fonctions complémentaires, la plupart de ces fonctions étant des primitives graphiques ou des fonctions de transformation de l'image ;
- les logiciels d'éducation et d'apprentissage du dessin pour les enfants ; dans la catégorie des programmes de jeux, citons pour mémoire le puzzle développé par Yves Lamoureux, qui est un très bon logiciel de manipulation d'une image ;
- enfin, le logiciel STARGRAPH est l'outil de dessin que nous avons utilisé pour fabriquer la série d'images sur le sport. Il a été développé conjointement par Daniel Fichter et Tera-Conseil.

ENVIRONNEMENT ET OUTILS DE DÉVELOPPEMENT

Sans entrer dans le détail – la documentation Amstrad et certains ouvrages spécialisés traitant de ces sujets (consultez les conseils de lecture en annexe) –, nous décrivons ci-après les moyens logiciels disponibles pour développer des applications, graphiques entre autres.

Le système d'exploitation des disquettes : (SED ou Disk Operating System)

Le système d'exploitation est un programme qui "gère" les informations entre la mémoire centrale et un support magnétique (disquette, cassette) ; il dispose d'un jeu d'instructions (commandes) utilisable depuis un programme Basic pour sauvegarder ou charger une image par exemple.

A la mise sous tension de l'Amstrad, le système par défaut est AMSDOS ; c'est le système que nous avons utilisé pour le développement des programmes. Les systèmes d'exploitation CP/M Plus et CP/M 2.2 sont fournis avec le modèle 6128 ; ils permettent d'exploiter au mieux les 128K disponibles, offrent par exemple la possibilité de gérer l'accès direct dans un fichier de données ; les ressources supplémentaires de CP/M n'étaient cependant pas indispensables pour nos travaux de gestion de l'image.

Le Basic Amstrad

C'est un langage de programmation évolué que nous utiliserons principalement pour gérer les "entrées-sorties" : chargement d'images, chargement et appel (call) des routines graphiques en assembleur.

Le Basic Amstrad est un langage interprété ; il n'est pas adapté à toutes les tâches que nous présentons dans ce livre : un tracé avec la souris (ou le joystick) – programmé en Basic – se révélerait trop lent ; d'autre part, les ordres graphiques disponibles ne permettent pas de réaliser les animations présentées dans la deuxième partie.

Les ordres graphiques du Basic sont les suivants (pour plus de précision sur le passage des paramètres, se référer à un document traitant du Basic) :

- ☐ BORDER : permet d'affecter une couleur au cadre extérieur de l'écran.
- ☐ PAPER : est la couleur de fond en mode texte.
- ☐ GRAPHICS PAPER : fixe la couleur d'encre du fond graphique.
- ☐ PEN : détermine la couleur du stylo, c'est-à-dire la couleur des caractères "texte".
- ☐ GRAPHICS PEN : fixe la couleur d'encre pour le tracé de lignes ou de points ; une valeur de mode "opaque" ou "transparent" peut également être spécifiée ; le mode transparent influe sur la couleur de fond d'un caractère écrit avec la commande TAG (voir plus loin).
- ☐ MASK : définit un masque de tracé de ligne : la valeur spécifiée (comprise entre 0 et 255) permet d'activer ou non un pixel dans un groupe de 8 pixels consécutifs. On peut ainsi tracer des lignes discontinues.
- ☐ INK : détermine la couleur d'une encre donnée ; 16 encres peuvent être spécifiées, et 27 couleurs sont disponibles :
  INK 1,23 signifie que l'encre n° 1 est turquoise pastel.
  INK 2,3,12 signifie que l'encre n° 2 est une couleur clignotante, alternativement rouge et jaune ; la vitesse de passage d'une couleur à une autre est définie par la commande SPEED INK.
- ☐ DRAW : trace une ligne entre la position courante du curseur graphique et un point absolu donné ; on spécifie également l'encre de tracé.

Un mode de tracé détermine l'interaction du tracé sur l'image en cours (4 modes sont disponibles : normal,XOR,AND,OR).
La fonction DRAWR est similaire, mais les coordonnées spécifiées (x,y) ne correspondent plus à un point absolu, mais à un déplacement par rapport à la position courante du curseur.

- ☐ PLOT : et PLOTR tracent des points.
- ☐ FILL : exécute un coloriage à partir de la position du curseur graphique ; la zone à colorier est quelconque ; un point appartient au contour s'il a la couleur de l'encre du stylo en cours ou celle de l'encre de fond.
- ☐ FRAME : gère la synchronisation de l'écriture de graphiques (caractères ou objets) sur l'image vidéo, et supprime donc l'effet de scintillement.
- ☐ TAG : (Text At Graphics) autorise l'insertion de texte sur une page graphique, la position du curseur graphique détermine la position du premier caractère de la chaîne "texte" ; on peut dans ce mode déplacer pixel par pixel des caractères ou symboles.
- ☐ TAGOFF : désactive la commande TAG.
- ☐ SYMBOL : permet de redéfinir un caractère du jeu Amstrad (222 caractères ou symboles codés de h'32 à h'255).
- ☐ MOVE : positionne le curseur graphique de manière absolue ; MOVER correspond à une translation par rapport à la dernière position.

Le jeu d'instructions Basic concernant les traitements graphiques est particulièrement riche, et facile à mettre en œuvre ; mais la programmation assembleur devient indispensable dès qu'il s'agit d'effectuer des transformations ou des "mouvements" d'images ; aussi, la quasi-totalité des modules listés dans ce livre sont écrits en assembleur.

Le langage Assembleur Z-80

Chaque micro-ordinateur possède un langage machine propre à son micro-processeur : le Z-80 dans le cas de l'Amstrad.

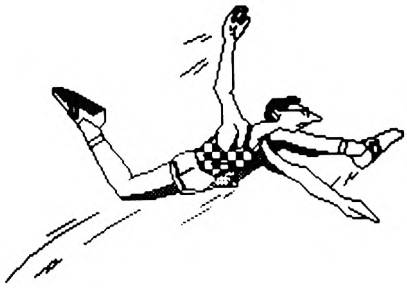
Le langage machine – seul langage exécutable par l'ordinateur – peut être généré par la traduction d'un programme assembleur.

Nous ne décrirons pas de manière exhaustive comment programmer en assembleur ; cependant, la lecture des routines proposées peut être un bon apprentissage.

Afin de faciliter la tâche du programmeur, il existe des outils (l'éditeur assembleur) destinés à la fabrication des sources assembleur et à la compilation de ces sources.

L'éditeur Assembleur

Nous avons utilisé l'éditeur-Assembleur de AMSOFT ; cet utilitaire est en fait un éditeur ligne, qui fort heureusement peut être remplacé par le pseudo-éditeur pleine page du Basic.



Chapitre 2
LA GESTION DE L'IMAGE

Les différents modes graphiques que nous allons décrire conditionnent la définition de l'image, quelles que soient les caractéristiques de résolution de l'écran de visualisation utilisé, et déterminent la palette de couleurs disponibles (ces couleurs sont créées par le mélange optique des trois couleurs fondamentales en électronique : le rouge, le vert et le bleu /RVB ou RGB en anglais).

Chaque point visualisé à l'écran est caractérisé par les informations suivantes :

- son adresse ou position à l'écran ;
- sa valeur de couleur.

Ces informations sont codées dans la mémoire ; la taille mémoire nécessaire étant conditionnée par la définition et le nombre de couleurs affichables sur une même image ; ainsi, compte tenu des limites de mémoire disponible, si l'on veut augmenter le nombre de couleurs, il faudra accepter une définition moindre.

LES MODES GRAPHIQUES ET LES SYSTÈMES DE COORDONNÉES

A la mise sous tension, votre Amstrad fonctionne par défaut en mode graphique "1" (ou mode 40 colonnes) ; le tableau suivant résume les caractéristiques des trois modes disponibles :

Mode	Résolution	Nb couleurs
0	160 * 200	16
1	320 * 200	4
2	640 * 200	2

La résolution verticale est la même, quel que soit le mode sélectionné ; les couleurs disponibles dans un mode sont choisies parmi une palette de 27 couleurs :

N° d'encre	Couleur
0	Noir
1	Bleu
2	Bleu vif
3	Rouge
4	Magenta
5	Mauve
6	Rouge vif
7	Violet
8	Magenta vif
9	Vert
10	Turquoise
11	Bleu ciel
12	Jaune
13	Blanc
14	Bleu pastel
15	Orange
16	Rose
17	Magenta pastel
18	Vert vif
19	Vert marin
20	Turquoise vif
21	Vert citron
22	Vert pastel
23	Turquoise pastel
24	Jaune vif
25	Jaune pastel
26	Blanc brillant

Une ligne écran est codée par 80 octets, quel que soit le mode sélectionné ; le tableau suivant donne, pour un mode graphique donné, le nombre de bits codant un pixel, et la correspondance bits/ pixel au sein d'un octet :

Mode	Résolution	Nb points/octet	Nb bits/pt
0	160 * 200	8	1
1	320 * 200	4	2
2	640 * 200	2	4

La correspondance bits / pixel est la suivante :

- mode "0" : 0 1 pixels
  1 5 3 7 0 4 2 6 bits
- mode "1" : 0 1 2 3 pixels
  3 7 2 6 1 5 0 4 bits
- mode "2" : 0 1 2 3 4 5 6 7 pixels
  7 6 5 4 3 2 1 0 bits

(le pixel 0 est le point le plus à gauche, le bit 7 est le poids le plus fort de l'octet).

Le mode "0" est celui que nous avons utilisé pour la fabrication des images de cet ouvrage et de la disquette d'accompagnement ; dans ce mode, 16 couleurs (INK) sont disponibles ; à l'initialisation de votre système, les valeurs par défaut de tracé (PEN) sont les suivantes :

N° du stylo (PEN)	N° d'encre (INK)
0	1
1	24
2	20
3	6
4	26
5	0
6	2
7	8
8	10
9	12
10	14
11	16
12	18
13	22
14	clignotement 1,24
15	clignotement 16,11

Nous avons utilisé cette palette par défaut dans le programme de création d'objets (*voir chapitre 3*) ; vous le constaterez sur la disquette d'accompagnement.

Mais cette affectation est bien entendu modifiable. La possibilité de faire clignoter des couleurs de tracé (et même de fond) est tout à fait intéressante pour réaliser des animations spécifiques ; une extension de cette possibilité "d'animation" est le cyclage des couleurs d'une image décrit dans la deuxième partie du livre.

LES SYSTÈMES DE COORDONNÉES

En Basic, l'utilisateur travaille toujours avec un écran idéal de résolution 640 par 200, le système se chargeant de convertir les coordonnées idéales (utilisateur) en coordonnées réelles suivant le mode graphique sélectionné.

En Assembleur, la majorité des routines systèmes disponibles fonctionnent en coordonnées "utilisateur", sauf celles qui sont spécifiques à un mode donné.

Trois systèmes de coordonnées sont disponibles :

- ☐ "Utilisateur" avec origine en bas et à gauche.
- ☐ "Utilisateur" avec origine redéfinie.
- ☐ Coordonnées réelles.

A l'exception d'une routine fonctionnant en coordonnées réelles, l'ensemble des programmes présentés fonctionne en mode "utilisateur" avec origine en bas et à gauche.

Un ensemble de routines systèmes permet de gérer la position courante du curseur graphique ; ces routines sont décrites en annexe, et accompagnées de commentaires dans les listes des programmes.

LA MÉMOIRE ÉCRAN

L'écran est représenté en mémoire par une zone de 16K débutant à l'adresse $C000 et finissant en $FFFF ; cette place correspond à la ROM supérieure contenant le Basic. Le système se charge de la commutation entre les deux plages ; aussi, lorsqu'on écrit à une adresse supérieure à $C000, on "écrit" automatiquement sur l'écran. Les routines systèmes qui affectent l'écran sont bien entendu contenues dans la ROM inférieure, mais possèdent un point d'entrée dans la RAM.

Description de l'écran

Il est divisé en 200 lignes de 80 octets ou en 25 "lignes texte" de 80 caractères (un caractère a une hauteur de 8 lignes).

Description de la mémoire écran

Elle est divisée en 8 blocks de 2K, le premier bloc contient la première ligne de chacune des 25 "lignes texte",..., le 8e bloc contient la dernière ligne de chacune des 25 "lignes texte ; cette implantation mémoire est résumée par le schéma suivant :

		<------- 80 octets ------->		
Ligne	1	C000	etc.	C04F
	2	C800		C84F
	3	D000		D04F
	4	D800		D84F
	5	E000		E04F
	6	E800		E84F
	7	F000		F04F
	8	F800		F84F
Ligne	9	C050		C09F
	193	C780		C7CF
	194	CF80		CFCF
	195	D780		D7CF
	196	DF80		DFCF
	197	E780		E7CF
	198	FF80		EFCF
	199	F780		F7CF
	200	FF80		FFCF

On remarque qu'à la fin de chaque block il y a 48 octets inutilisés pour la description de l'image : en effet, un block a une taille de 2 048 octets et contient 25 lignes de 80 octets (soit 2 000 octets). Ces octets sont en fait utilisés pour effectuer des "déroulements" d'écran (provoqués par exemple par un "PRINT" ou <ENTER> au-delà de la dernière

ligne) ; un défilement d'écran consiste à modifier la valeur de départ de l'écran : cette valeur de départ vaut $C000 + offset ; l'offset est une translation comprise entre $0 et $7FF.

L'activation d'une commande MODE remet à zéro l'offset ; les programmes contenus dans cet ouvrage sont initialisés par un "MODE 0" ; la mémoire écran commence donc toujours en $C000.

COMMENT FONCTIONNE LA COULEUR ?

Vous avez remarqué que les couleurs fondamentales en électronique sont le rouge, le vert et le bleu alors qu'en peinture le vert est obtenu par mélange du bleu et du jaune (le jaune étant une couleur primaire) ; une gamme de couleur est obtenue par mélange des couleurs primaires : le mélange est dit additif en électronique (soustractif en peinture), le mélange parfait des trois couleurs de base donnant du blanc (le mélange des pigments primaires conduisant invariablement à du marron foncé...).

Il existe deux technologies pour reproduire la couleur :

- l'intérieur de la surface d'écran de votre moniteur vidéo est constitué d'une couche de phosphore ; en réalité, cette couche n'est pas uniforme, elle est constituée de "triades" de phosphore – chaque point de la triade étant rouge, vert et bleu –. Le tube cathodique est constitué de trois canons à électrons (R/V/B) ; les électrons émis par chaque canon viennent frapper le point de phosphore correspondant après avoir traversé une grille (masque perforé).
- l'autre technique est celle du tube à pénétration : la couleur est produite par un canon unique ; les électrons frappant plusieurs couches de phosphore : en faisant varier la tension d'accélération du faisceau d'électrons, on agit sur la pénétration dans les couches et on visualise donc des couleurs différentes.

En fait, dans la première technique (utilisée dans les moniteurs bon marché), les points de la triade sont si rapprochés qu'il sera impossible de percevoir les trois composantes et seule la couleur résultante sera perçue.

Mais la perception que nous avons des couleurs n'est pas toujours conforme avec "la réalité des faisceaux émis" ; ainsi deux triades consécutives interagissent : si vous fabriquez une trame "rouge et bleu" par exemple, et coloriez une partie de votre image à l'aide de cette trame, la surface coloriée apparaîtra violette ; l'examen des points à la loupe ne décèlera aucun point violet ; l'utilisation de trames enrichit donc artificiellement la palette puisque nous n'y voyons que du feu !

Chapitre 3

LE DEPLACEMENT D'OBJETS

Pour dessiner à l'écran, nous devons apprendre à représenter un outil de dessin schématisé, à visualiser le tracé effectué par cet outil, et à gérer son déplacement à l'aide d'un joystick ou d'une souris en l'occurrence.

LA FABRICATION DE FORMES OU D'OBJETS

La possibilité de construire et d'animer des formes est une ressource importante ; l'intérêt de l'utilisation des formes est très varié :

- une forme peut être l'élément d'un jeu d'adresse : un monstre, un personnage, une fusée, etc. ; le célèbre "Pacman", par exemple, est une forme qui a la forme ;
- une forme peut être un motif répétitif utilisé dans la fabrication d'une image ; il peut être intéressant de se constituer une bibliothèque de formes appelables à tout instant.
  Dans un logiciel de décoration intérieure, par exemple, une table de formes sera constituée d'éléments de mobilier, sanitaires, lampes, plantes etc. ;
- on enregistre dans une table de formes l'ensemble des caractères constituant une police d'un style particulier ;

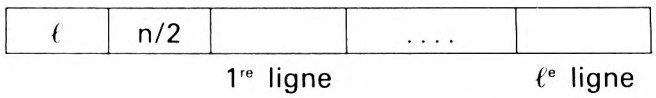
- une forme est aussi un objet symbolique visualisant une fonction précise, comme le crayon que nous allons déplacer avec le joystick.

Plusieurs solutions de description de la forme en mémoire sont possibles :

- la forme est un caractère redéfini par l'utilisateur ;
- une zone mémoire décrit la forme point par point, la taille de la zone est égale à la taille de l'objet ;
- la forme est enregistrée comme un ensemble de vecteurs (ou une liste de commandes plus évoluées : arcs, cercles...) ; un programme interpréteur est nécessaire pour l'affichage.

Nous vous présentons les deux premières possibilités :

- une forme "caractère" est gérée par la commande SYMBOL du Basic ; une restriction cependant, notre forme aura une couleur unique ;
- l'objet est décrit point par point, il est contenu dans un rectangle de largeur "n" points et de hauteur "ℓ" lignes ; la zone mémoire peut être ainsi schématisée :

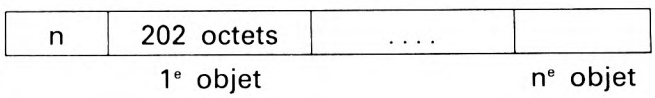


Une ligne contient n/2 octets ; en effet, en mode "0", un octet contient 2 points (si n est impair, on prendra l'arrondi supérieur de n/2.

L'utilisation de cette méthode permet de dessiner l'objet en 16 couleurs, la place mémoire nécessaire est bien entendu plus importante.

Nous décrivons ci-après le programme Basic CREOBJ ; ce programme permet de créer, visualiser ou modifier les éléments d'une table d'objets TABOBJ.BIN.

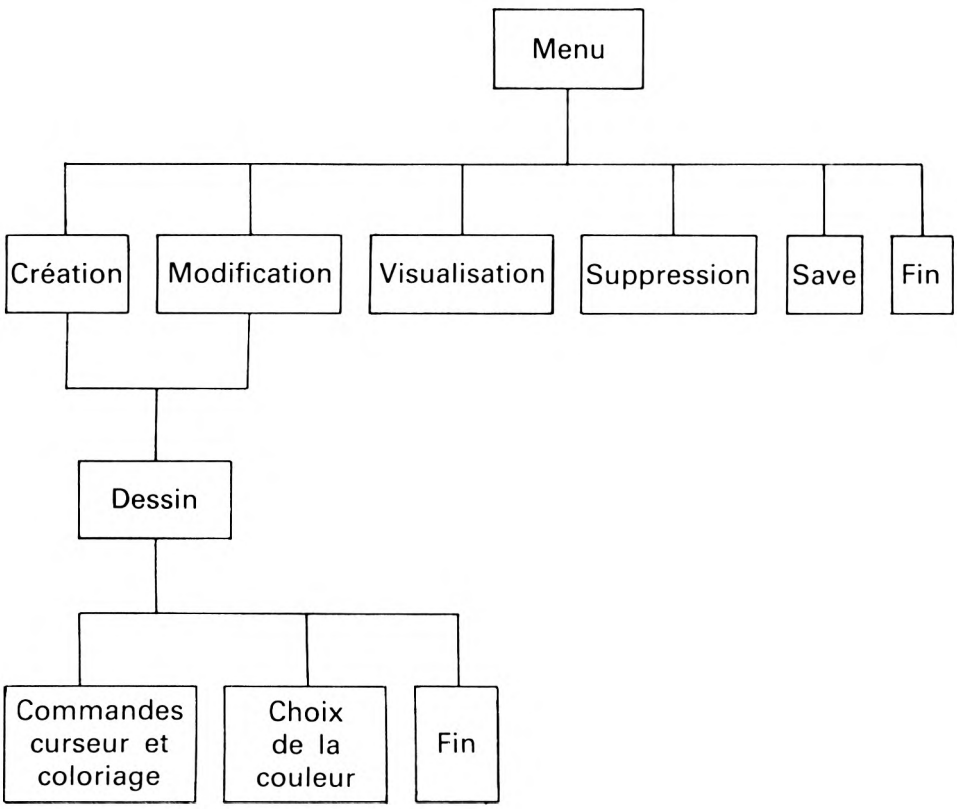
Cette table d'objets a la structure suivante :



L'objet fabriqué est stocké dans une zone de longueur fixe de 202 octets, il est inscrit dans un rectangle ayant une taille maximale de 20 points par 20 lignes ; 10 octets permettent de coder 20 points,

200 octets sont donc nécessaires pour coder le rectangle, les deux octets complémentaires servent à enregistrer la largeur et la hauteur réelle de l'objet inscrit dans le rectangle.

L'algorithme général du **programme CREOBJ.BAS** est le suivant :



Voici la liste du **programme CREOBJ** :

```
10 '        ********    CREATION D'OBJETS    ********
20 '
30 MEMORY 18999:ON ERROR GOTO 1980
40 LOAD "affcre.ass",19200:LOAD "tabobj",22160:nbobj=PEEK(22160)
50 'AFFICHAGE MENU
60 '--------------
70 MODE 1:PRINT TAB(11);"CREATION D'OBJETS"
80 RESTORE 130
```

```
90 FOR i=1 TO 6:LOCATE 1,2*i+4:READ a$:PRINT i;"- ";a$:NEXT i
100 GOSUB 1810
110 CLEAR INPUT:ON b GOSUB 760,140,570,420,280,350
120 GOTO 50
130 DATA Creation,Visualisation,Modification,Suppression,Sauvegarde,Fin
140 'VISUALISATION
150 '-------------
160 IF nbobj=0 THEN RETURN
170 MODE 0:PRINT TAB(3);"VISUALISATION"
180 FOR i=1 TO nbobj
190    LOCATE 2-7*(i>5),5+3*(i+5*(i>5)):PRINT i
200    MOVE 160-200*(i>5),343-48*(i+5*(i>5))
210    adr=22161+202*(i-1)
220    POKE 19201,adr-INT(adr/256)*256:POKE 19202,INT(adr/256)
230    CALL 19200
240 NEXT i
250 IF INKEY$<>"" THEN 250
260 IF INKEY$="" THEN 260
270 RETURN
280 'SAUVEGARDE
290 '----------
300 MODE 1:PRINT "Confirmation sauvegarde ? (O/N)"
310 a$=INKEY$:IF a$="" THEN 310
320 IF a$<>"O" AND a$<>"o" THEN RETURN
330 SAVE "TABOBJ",b,22160,1+202*nbobj
340 RETURN
350 'FIN
360 '---
370 MODE 1:PRINT TAB(18);"FIN"
380 LOCATE 1,8:PRINT "Sauvegarde ? (O/N)"
390 a$=INKEY$:IF a$="" THEN 390
400 IF a$="o" OR a$="O" THEN SAVE "TABOBJ",b,22160,1+202*nbobj
410 RUN "menu"
420 'SUPPRESSION
430 '-----------
440 IF nbobj=0 THEN RETURN
450 MODE 0:PRINT TAB(4);"SUPPRESSION"
460 GOSUB 180
470 LOCATE 1,24:INPUT "Numero de l'objet ";d
480 IF d=0 THEN RETURN
490 IF d<0 OR d>nbobj THEN 470
500 FOR j=d+1 TO nbobj
510    FOR k=0 TO 273
520       POKE ((j-2)*202+22161+k),PEEK((j-1)*202+22161+k)
530    NEXT k
540 NEXT j
550 nbobj=nbobj-1:POKE 22160,nbobj
560 RETURN
570 'MODIFICATION
```

```
580 '------------
590 IF nbobj=0 THEN RETURN
600 MODE 0:PRINT TAB(3);"MODIFICATION"
610 GOSUB 180
620 LOCATE 1,23:INPUT "Numero de l'objet ";m
630 IF m=0 THEN RETURN
640 IF m<0 OR m>nbobj THEN 620
650 h=PEEK(22161+(m-1)*202):l=(PEEK(22162+(m-1)*202))*2:ptr=22163+(m-1)*202
660 GOSUB 850:GOSUB 1740
670 FOR y=4+(h-1)*14 TO 4 STEP -14
680    FOR x=8 TO 8+(l-2)*28 STEP 56
690       POKE 19255,PEEK(ptr):CALL 19278
700       MOVE x,y:FILL PEEK(19253)
710       MOVE x+28,y:FILL PEEK(19254)
720       ptr=ptr+1
730    NEXT x
740 NEXT y
750 x=8:y=4:GOSUB 1740:GOTO 1080
760 'CREATION
770 '--------
780 IF nbobj=10 THEN RETURN ELSE nbobj=nbobj+1:m=nbobj:POKE 22160,m
790 MODE 1:PRINT TAB(16);"CREATION"
800 LOCATE 1,4:INPUT "Largeur (entre 1 et 20) ";l
810 IF l<1 OR l>20 THEN 800
820 LOCATE 1,7:INPUT "Hauteur (entre 1 et 20) ";h
830 IF h<1 OR h>20 THEN 820
840 GOSUB 850:GOTO 1080
850 'AFFICHAGE COMMANDES
860 '------------------
870 MODE 1:RESTORE 940:PRINT TAB(15);"COMMANDES":LOCATE 1,4
880 FOR i=1 TO 14
890    READ a,b,a$:IF i>11 THEN PRINT TAB(6);
900 IF a=1 THEN PRINT " ";
910 PRINT CHR$(a);CHR$(b);" : ";a$
920 NEXT i
930 IF INKEY$="" THEN 930
940 DATA 1,11, ,1,10,deplacement curseur,1,8, ,1,9,
950 DATA 102,56,coloriage haut colonne,102,50,coloriage bas colonne
960 DATA 102,52,coloriage debut ligne,102,54,coloriage fin ligne
970 DATA 1,13,coloriage case,102,32,fin dessin,99,32,choix couleur
980 DATA 1,8,deplacement curseur,1,9, ,1,13,fin choix couleur
990 'AFFICHAGE ECRAN POUR DESSIN
1000 '--------------------------
1010 MODE 0:WINDOW #1,1,20,22,25:ORIGIN 0,63
1020 FOR y=0 TO h*14 STEP 14:MOVE 0,y:DRAW l*28,y,1,0:NEXT y
1030 FOR x=0 TO l*28 STEP 28:MOVE x,0:DRAW x,h*14,1,0:NEXT x
1040 x=8:y=4:GOSUB 1740
1050 LOCATE #1,2,2:FOR i=0 TO 15:PEN #1,i:PRINT #1,CHR$(143);:NEXT i
1060 c=3:LOCATE #1,c,3:PEN #1,1:PRINT #1,CHR$(240)
```

```
1070 RETURN
1080 'DESSIN D'UN OBJET
1090 '-----------------
1100 GOSUB 1820
1110 ON b GOTO 1620,1470,1290,1120,1160,1200,1240,1310,1350,1390,1430
1120 'HAUT
1130 '----
1140 IF y+14<h*14 THEN GOSUB 1740:y=y+14:GOSUB 1740
1150 GOTO 1080
1160 'BAS
1170 '---
1180 IF y-14>0 THEN GOSUB 1740:y=y-14:GOSUB 1740
1190 GOTO 1080
1200 'GAUCHE
1210 '------
1220 IF x-28>0 THEN GOSUB 1740:x=x-28:GOSUB 1740
1230 GOTO 1080
1240 'DROITE
1250 '------
1260 IF x+28<l*28 THEN GOSUB 1740:x=x+28:GOSUB 1740
1270 GOTO 1080
1280 'COLORIAGE CASE
1290 '--------------
1300 GOSUB 1740:FILL c-2:GOSUB 1740:GOTO 1080
1310 'COLORIAGE HAUT COLONNE
1320 '----------------------
1330 GOSUB 1740:FOR yc=y TO h*14-10 STEP 14:MOVE x,yc:FILL c-2:NEXT yc
1340 GOSUB 1740:GOTO 1080
1350 'COLORIAGE BAS COLONNE
1360 '----------------------
1370 GOSUB 1740:FOR yc=y TO 4 STEP -14:MOVE x,yc:FILL c-2:NEXT yc
1380 GOSUB 1740:GOTO 1080
1390 'COLORIAGE DEBUT LIGNE
1400 '---------------------
1410 GOSUB 1740:FOR xc=x TO 8 STEP -28:MOVE xc,y:FILL c-2:NEXT xc
1420 GOSUB 1740:GOTO 1080
1430 'COLORIAGE FIN LIGNE
1440 '-------------------
1450 GOSUB 1740:FOR xc=x TO l*28-20 STEP 28:MOVE xc,y:FILL c-2:NEXT xc
1460 GOSUB 1740:GOTO 1080
1470 'CHOIX DE LA COULEUR
1480 '-------------------
1490 GOSUB 1740
1500 GOSUB 1830:ON b GOTO 1590,1540,1510
1510 'COULEUR GAUCHE
1520 '--------------
1530 IF c=2 THEN 1500 ELSE nc=c-1:GOTO 1570
1540 'COULEUR DROITE
1550 '--------------
```

```
1560 IF c=17 THEN 1500 ELSE nc=c+1
1570 LOCATE #1,c,3:PRINT #1," ":c=nc:LOCATE #1,c,3:PRINT #1,CHR$(240)
1580 GOTO 1500
1590 'FIN CHOIX COULEUR
1600 '-----------------
1610 GRAPHICS PEN c-2:GOSUB 1740:GOTO 1080
1620 'FIN DESSIN OBJET
1630 '----------------
1640 GOSUB 1740:cpt=22161+(m-1)*202
1650 POKE cpt,h:POKE cpt+1,INT(l/2+0.5):cpt=cpt+2
1660 FOR y=h*14-10 TO 4 STEP -14
1670    FOR x=8 TO l*28-20 STEP 56
1680       z=TEST(x,y):POKE 19253,z
1690       z=TEST(x+28,y):POKE 19254,z
1700       CALL 19256:POKE cpt,PEEK(19255):cpt=cpt+1
1710    NEXT x
1720 NEXT y
1730 RETURN
1740 'ECRITURE / EFFACEMENT CURSEUR
1750 '----------------------------
1760 MOVE x,y
1770 DRAWR 12,0,1,1:DRAWR 0,6,1,1:DRAWR -12,0,1,1:DRAWR 0,-6,1,1
1780 RETURN
1790 'ATTENTE D'UNE COMMANDE
1800 '----------------------
1810 fin=6:b=0:WHILE(b=0):RESTORE 1920:GOSUB 1840:WEND:RETURN
1820 fin=11:b=0:WHILE(b=0):RESTORE 1940:GOSUB 1840:WEND:RETURN
1830 fin=3:b=0:WHILE(b=0):RESTORE 1960:GOSUB 1840:WEND:RETURN
1840 b=0
1850 FOR i=1 TO fin
1860    READ t
1870    IF INKEY(t)=0 THEN b=i:i=fin
1880 NEXT i
1890 RETURN

1900 'CODES DES TOUCHES DE COMMANDE
1910 '-----------------------------
1920 DATA 64,65,57,56,49,48
1940 DATA 53,62,18,0,2,8,1,11,14,20,4
1960 DATA 18,1,8
1980 IF DERR=146 THEN nbobj=0:POKE 22160,0:RESUME NEXT
1990 PRINT "erreur ligne ";ERL;"  ERR=";ERR;"DERR=";DERR
2000 a$=INKEY$:IF a$="" THEN 2000
2010 PRINT a$;:GOTO 2000
```

```
                    1 ; -------- AFFICHAGE OBJET --------
                    2 ;
4B00                3         ORG  19200
4B00  21C05D        4         LD   HL,24000          ;debut objet
4B03  46            5         LD   B,(HL)            ;hauteur objet
4B04  23            6         INC  HL
4B05  4E            7         LD   C,(HL)            ;largeur objet
4B06  23            8         INC  HL
4B07  EB            9         EX   DE,HL             ;DE=debut dessin objet
4B08  CD1C4B       10         CALL ADR               ;HL=adresse ecran curs.
4B0B  EB           11         EX   DE,HL             ;echange DE-HL
                   12 ;
4B0C  C5           13 BCL1:   PUSH BC
4B0D  D5           14         PUSH DE
4B0E  0600         15         LD   B,0               ;BC=largeur objet
4B10  EDB0         16         LDIR                   ;transfert ligne
4B12  D1           17         POP  DE
4B13  EB           18         EX   DE,HL
4B14  CD26BC       19         CALL #BC26             ;ligne ecran suivante
4B17  EB           20         EX   DE,HL
4B18  C1           21         POP  BC
4B19  10F1         22         DJNZ BCL1              ;test fin objet
4B1B  C9           23         RET
                   24 ;
                   25 ; Conversion position curseur graphique->adresse ecran:
                   26 ;
4B1C  F5           27 ADR:    PUSH AF
4B1D  C5           28         PUSH BC
4B1E  D5           29         PUSH DE
4B1F  CDC6BB       30         CALL #BBC6             ;HL,DE=coordonnees curs.
4B22  CB3C         31         SRL  H                 ;Y / 2
4B24  CB1D         32         RR   L
4B26  CB3A         33         SRL  D                 ;X / 4
4B28  CB1B         34         RR   E
4B2A  CB3A         35         SRL  D
4B2C  CB1B         36         RR   E
4B2E  CD1DBC       37         CALL #BC1D             ;=>HL=adresse ecran
4B31  D1           38         POP  DE
4B32  C1           39         POP  BC
4B33  F1           40         POP  AF
4B34  C9           41         RET
                   42 ;
                   43 ; ----  CONVERSION 2 N0 ENCRES -> ENCRES CODEES SUR 1 OCTET
                   44 ;
4B35  00           45 ENCG:   DEFB 0                 ;N0 encre pt gauche
4B36  00           46 ENCD:   DEFB 0                 ;N0 encre pt droite
4B37  00           47 OCTET:  DEFB 0                 ;resultat code
4B38  3A354B       48         LD   A,(ENCG)
4B3B  CD2CBC       49         CALL #BC2C             ;A=masque encre pt gauche
4B3E  E6AA         50         AND  %10101010         ;effacement pt droite du masque
4B40  47           51         LD   B,A
```

```
4B41  3A364B      52        LD   A,(ENCD)
4B44  CD2CBC      53        CALL #BC2C              ;A=masque encre pt droite
4B47  E655        54        AND  %01010101          ;effacement pt gauche du masque
4B49  B0          55        OR   B                  ;A=pt gauche,pt droite
4B4A  32374B      56        LD   (OCTET),A
4B4D  C9          57        RET
                  58 ;
                  59 ; ---- CONVERSION INVERSE DE LA PRECEDENTE ----
                  60 ;
4B4E  3A374B      61        LD   A,(OCTET)
4B51  E6AA        62        AND  %10101010          ;masquage pt gauche
4B53  CD2FBC      63        CALL #BC2F              ;decodage encre
4B56  32354B      64        LD   (ENCG),A
                  65 ;
4B59  3A374B      66        LD   A,(OCTET)
4B5C  E655        67        AND  %01010101          ;masquage pt droite
4B5E  17          68        RLA                     ;devient pt gauche
4B5F  CD2FBC      69        CALL #BC2F              ;decodage encre
4B62  32364B      70        LD   (ENCD),A
4B65  C9          71        RET
```

Remarques sur le programme CREOBJ.BAS

Le module VISUALISATION fait appel à une routine assembleur (19200) pour afficher à la position du curseur graphique un objet dont l'adresse de départ est stockée en 19201 et 19202.

La fonction SUPPRESSION détruit l'objet sélectionné ; cet objet est écrasé par son suivant immédiat, le processus se répétant jusqu'à la fin de la table d'objets.

La routine MODIFICATION utilise une routine assembleur (19278) de décodage d'un "octet-objet" (octet stocké en 19255) en deux numéros de couleurs correspondant aux points gauche et droite (situés en 19253 et 19254 respectivement).

Le module AFFICHAGE ÉCRAN POUR DESSIN trace une grille de description de l'objet point par point.

La gestion des commandes est détaillée en annexe A1 (guide utilisateur de la disquette).

Le module FIN fait appel à un programme assembleur (19256) combinant deux numéros de couleur (19253/point gauche et 19254/ point droite) en un "octet-objet" situé en 19255 et représentant deux pixels en mode "0".

Le curseur de repérage (mode dessin) est tracé en mode graphique XOR ; il est donc effacé par surimpression.

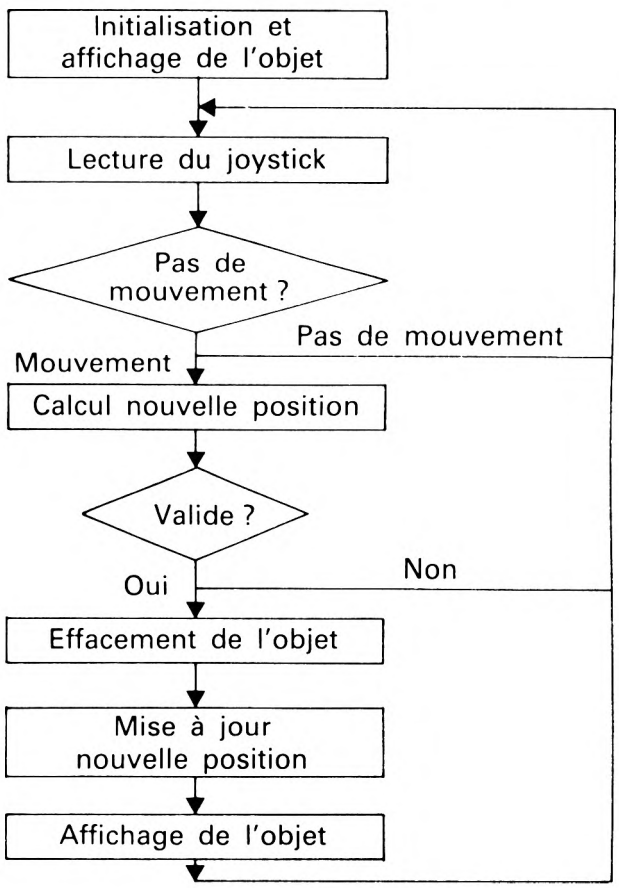
Signalons enfin que les points coloriés avec la couleur n° 1 ne peuvent être effacés ; pour les autres couleurs, il n'y a pas de contraintes.

Les routines Assembleur appelées dans ce programme sont regroupées dans le programme AFFCRE.ASS. Certaines instructions utilisées étant spécifiques aux modèles CPC 664 et 6128, ce programme nécessite quelques modifications en vue d'une adaptation pour le CPC 464 : l'instruction FILL. par exemple, peut être remplacée par une suite de tracés de droites (la zone à colorier étant un rectangle).

LA GESTION DU MOUVEMENT D'UNE FORME OU OBJET

Nous venons de créer nos premiers objets, il s'agit maintenant de programmer leur déplacement à l'écran.

L'algorithme général du déplacement



Voici la liste du module **Assembleur DEPLF.ASS** (ce module fait appel aux sous-programmes de validation de la position – VALID.ASS – et d'affichage de l'objet – AFFOBJ.ASS – décrit dans les pages suivantes).

```
                    1 ; -------- DEPLACEMENT D'UN OBJET --------
                    2 ;
                    3 ;            PROGRAMME PRINCIPAL
                    4 ;
4A38                5          ORG  19000
4A38  1803          6          JR   DEBUT
4A3A  01            7 OBJ:     DEFB 1
4A3B  9156          8 ADROBJ:  DEFW 22161
4A9C                9 VALID:   EQU  19100
4B00               10 AFFOBJ:  EQU  19200
4A3D  3A3A4A       11 DEBUT:   LD   A,(OBJ)          ;n0 de l'objet
4A40  21C755       12          LD   HL,21959         ;adr 1er objet - 202
4A43  11CA00       13          LD   DE,202           ;longueur objet
4A46  47           14          LD   B,A
4A47  19           15 SUIVAN:  ADD  HL,DE            ;HL=adresse objet
4A48  10FD         16          DJNZ SUIVAN
4A4A  223B4A       17          LD   (ADROBJ),HL      ;stockage adresse
4A4D  CD004B       18          CALL AFFOBJ           ;affichage objet
                   19 ;
4A50  3E42         20 RELECT:  LD   A,66              ;code touche ESC
4A52  CD1EBB       21          CALL #BB1E            ;test touche
4A55  C0           22          RET  NZ               ;retour si appuyee
4A56  CD24BB       23          CALL #BB24            ;lecture joystick
4A59  E60F         24          AND  #0F
4A5B  28F3         25          JR   Z,RELECT         ;relecture si rien
                   26 ;
4A5D  CD9C4A       27          CALL VALID            ;deplacement
4A60  38EE         28          JR   C,RELECT         ;relecture si deplacement
                   29 ;                               impossible
4A62  E5           30          PUSH HL               ;sauv. nouvelle position
4A63  D5           31          PUSH DE
4A64  CD19BD       32          CALL #BD19            ;synchronisation
4A67  2A3B4A       33          LD   HL,(ADROBJ)
4A6A  CD004B       34          CALL AFFOBJ           ;effacement objet
4A6D  D1           35          POP  DE               ;rest. nouvelle position
4A6E  E1           36          POP  HL
4A6F  CDC0BB       37          CALL #BBC0            ;mise a jour pos.curseur
4A72  CD19BD       38          CALL #BD19            ;synchronisation
4A75  2A3B4A       39          LD   HL,(ADROBJ)
4A78  CD004B       40          CALL AFFOBJ           ;affichage objet
4A7B  18D3         41          JR   RELECT
```

Le calcul et la validation de la nouvelle position

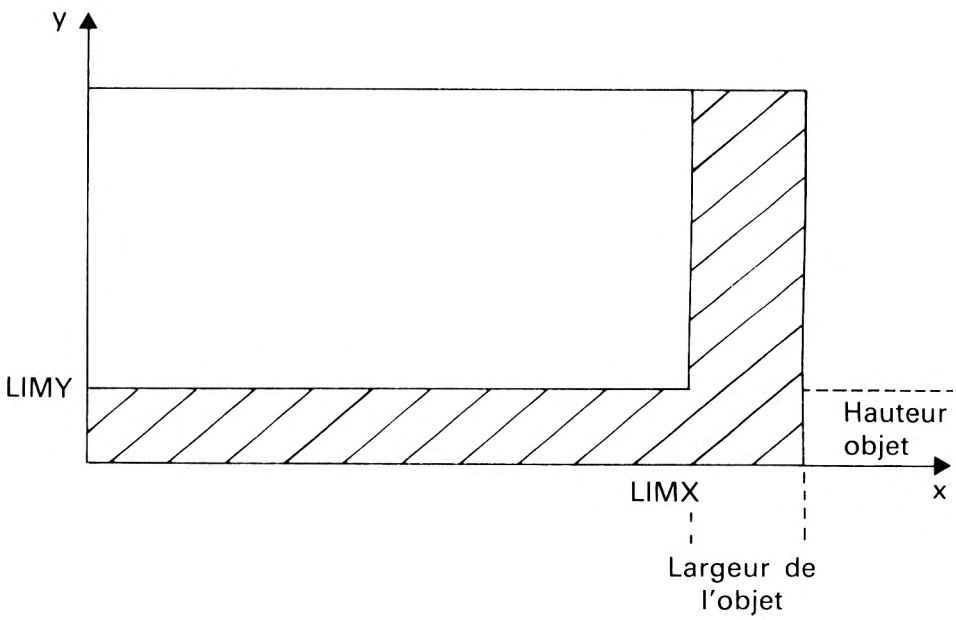
Cette fonction est gérée par le programme assembleur VALID.ASS : les nouvelles coordonnées du curseur graphique sont déduites de "l'état" du joystick ; pour être validée, la nouvelle position doit autoriser une visualisation complète de l'objet.

Le déplacement est calculé par "pas" ou groupe de points (PASX et PASY) ; puisque la position du curseur est définie en "coordonnées idéales" (640 par 400), les pas "X" et "Y" doivent l'être également.

En mode "0", on aura donc :

- PASX = 8 <===> déplacement de 2 points horizontalement ;
- PASY = 2 <===> déplacement d'1 point verticalement.

La position du curseur correspond au point en haut et à gauche de l'objet, aussi la visualisation complète de l'objet détermine les contraintes d'affichage ainsi schématisées :



(la zone hachurée correspond à la zone interdite pour le curseur graphique).

Voici la liste du **programme Assembleur VALID.ASS :**

```
                    1 ; -------- TEST VALIDITE DEPLACEMENT --------
                    2 ;
                    3 ;Entree: A=etat joystick
                    4 ;
                    5 ;Sortie: si Cy=1 deplacement impossible
                    6 ;        si Cy=0 deplacement possible
                    7 ;                et HL,DE = nouvelles coordonnes
                    8 ;
4A9C                9          ORG  19100
4A9C  1808         10          JR   VAL
4A9E  0400         11 PASX:    DEFW 4                  ;pas X
4AA0  0200         12 PASY:    DEFW 2                  ;pas Y
4AA2  4002         13 LIMX:    DEFW 576                ;limite superieure X
4AA4  2000         14 LIMY:    DEFW 32                 ;limite inferieure Y
4AA6  F5           15 VAL:     PUSH AF
4AA7  CDC6BB       16          CALL #BBC6              ;HL,DE=coordonnees curs.
4AAA  F1           17          POP  AF
4AAB  1F           18 HAUT:    RRA                     ;Cy=etat joystick haut
4AAC  300F         19          JR   NC,BAS
4AAE  ED4BA04A     20          LD   BC,(PASY)
4AB2  09           21          ADD  HL,BC              ;nouvelle coord. Y
4AB3  E5           22          PUSH HL
4AB4  019001       23          LD   BC,400             ;limite sup Y
4AB7  B7           24          OR   A
4AB8  ED42         25          SBC  HL,BC              ;Y < limite sup ?
4ABA  E1           26          POP  HL
4ABB  3F           27          CCF
4ABC  D8           28          RET  C                  ;retour si non
4ABD  1F           29 BAS:     RRA                     ;Cy=etat joystick bas
4ABE  3011         30          JR   NC,GAUCHE
4AC0  ED4BA04A     31          LD   BC,(PASY)
4AC4  B7           32          OR   A
4AC5  ED42         33          SBC  HL,BC              ;nouvelle coord. Y
4AC7  D8           34          RET  C                  ;retour si Y < 0
4AC8  E5           35          PUSH HL
4AC9  ED4BA44A     36          LD   BC,(LIMY)
4ACD  ED42         37          SBC  HL,BC              ;Y > limite inf ?
4ACF  E1           38          POP  HL
4AD0  D8           39          RET  C                  ;retour si non
4AD1  EB           40 GAUCHE:  EX   DE,HL
4AD2  1F           41          RRA                     ;Cy=etat joystick gauche
4AD3  3008         42          JR   NC,DROITE
4AD5  ED4B9E4A     43          LD   BC,(PASX)
4AD9  B7           44          OR   A
4ADA  ED42         45          SBC  HL,BC              ;nouvelle coordonnee X
```

```
4ADC  D8          46          RET  C                ;retour si X < 0
4ADD  1F          47 DROITE:  RRA                   ;Cy=etat joystick droite
4ADE  3010        48          JR   NC,FIN
4AE0  ED4B9E4A    49          LD   BC,(PASX)
4AE4  09          50          ADD  HL,BC            ;nouvelle coordonnee X
4AE5  E5          51          PUSH HL
4AE6  ED4BA24A    52          LD   BC,(LIMX)
4AEA  B7          53          OR   A
4AEB  ED42        54          SBC  HL,BC            ;X < limite sup ?
4AED  E1          55          POP  HL
4AEE  3F          56          CCF
4AEF  D8          57          RET  C                ;retour si non
4AF0  EB          58 FIN:     EX   DE,HL
4AF1  C9          59          RET
```

L'affichage et l'effacement de l'objet (méthode de l'empreinte)

Dans cette méthode, le tracé de l'objet est réalisé par un "ou exclusif" (XOR) entre la zone écran et la zone écran objet.

Les avantages de cette méthode sont les suivants :

- la zone mémoire utilisée est minimale ;
- deux affichages consécutifs (c'est-à-dire à la même adresse) restaurent l'image initiale : en effet, cela correspond au traitement (P XOR Q) XOR Q, P étant la valeur du fond et Q celle de la zone objet ; or on a l'égalité : (P XOR Q) XOR Q = P ;
- rapidité de traitement.

L'inconvénient est que les couleurs de l'objet changent suivant la nature du fond sur lequel il se déplace.

Deux modules Assembleur ont été étudiés (ces deux modules sont indépendants) :

- le premier module **XOR1.ASS** résout le cas où le pas de déplacement PASX est pair ; dans ce cas, l'objet est superposé tel quel à l'écran ;
- le second module **XOR2.ASS** traite les cas où PASX est soit impair, soit pair ; si l'abscisse (départ de l'objet) d'affichage est paire, on est ramené au cas précédent ; par contre, si cette abscisse est impaire, il y a décalage de l'objet par rapport à l'écran : le point "gauche" de chaque octet de l'objet doit être superposé au point de droite de l'octet-écran correspondant, et le point "droite" doit être superposé au point gauche de l'octet-écran correspondant suivant.

Voici les listes des deux **modules XOR1.ASS et XOR2.ASS :**

```
                    1 ; -------- SURIMPRESSION "XOR" --------
                    2 ;
                    3 ;-> OU EXCLUSIF (XOR) entre zone memoire objet et ecran
                    4 ;
                    5 ;(cas ou le pas de deplacement suivant x est pair)
                    6 ;
                    7 ;Entree: HL=adresse de l'objet
                    8 ;
4B00                9         ORG  19200
4B00  4E           10         LD   C,(HL)            ;hauteur objet
4B01  23           11         INC  HL
4B02  46           12         LD   B,(HL)            ;largeur objet
4B03  23           13         INC  HL
4B04  EB           14         EX   DE,HL             ;DE=debut dessin objet
4B05  CD1A4B       15         CALL ADR               ;HL=adresse ecran
                   16 ;
4B08  C5           17 BCL1:   PUSH BC
4B09  E5           18         PUSH HL
                   19 ;
4B0A  1A           20 BCL2:   LD   A,(DE)
4B0B  AE           21         XOR  (HL)
4B0C  77           22         LD   (HL),A            ;(HL)=(HL) XOR (DE)
4B0D  23           23         INC  HL
4B0E  13           24         INC  DE
4B0F  10F9         25         DJNZ BCL2              ;test fin ligne
                   26 ;
4B11  E1           27         POP  HL
4B12  CD26BC       28         CALL #BC26             ;ligne ecran suivante
4B15  C1           29         POP  BC
4B16  0D           30         DEC  C
4B17  20EF         31         JR   NZ,BCL1           ;test fin objet
4B19  C9           32         RET
                   33 ;
                   34 ; Conversion coordonnees curs. graphique -> adresse ecran
                   35 ;
4B1A  F5           36 ADR:    PUSH AF
4B1B  C5           37         PUSH BC
4B1C  D5           38         PUSH DE
4B1D  CDC6BB       39         CALL #BBC6             ;HL,DE=coord. curseur
                   40 ;
4B20  CB3C         41         SRL  H                 ;Y / 2
4B22  CB1D         42         RR   L
                   43 ;
4B24  CB3A         44         SRL  D                 ;X / 4
4B26  CB1B         45         RR   E
```

```
4B28  CB3A      46          SRL  D
4B2A  CB1B      47          RR   E
                48 ;
4B2C  CD1DBC    49          CALL #BC1D              ;-> adresse ecran
4B2F  D1        50          POP  DE
4B30  C1        51          POP  BC
4B31  F1        52          POP  AF
4B32  C9        53          RET


                 1 ; --------  SURIMPRESSION "XOR"  --------
                 2 ;
                 3 ;-> OU EXCLUSIF (XOR) entre zone memoire objet et ecran
                 4 ;
                 5 ;(cas ou le pas de deplacement suivant x est impair)
                 6 ;
                 7 ;Entree: HL=adresse de l'objet
                 8 ;
4B00             9          ORG  19200
4B00  4E        10          LD   C,(HL)             ;hauteur objet
4B01  23        11          INC  HL
4B02  46        12          LD   B,(HL)             ;largeur objet
4B03  23        13          INC  HL
4B04  EB        14          EX   DE,HL              ;DE=debut dessin objet
4B05  C5        15          PUSH BC
4B06  D5        16          PUSH DE
4B07  CDC6BB    17          CALL #BBC6              ;HL,DE=coord. curseur
                18 ;
4B0A  CB3C      19          SRL  H                  ;Y / 2
4B0C  CB1D      20          RR   L
                21 ;
4B0E  CB3A      22          SRL  D                  ;X / 4
4B10  CB1B      23          RR   E
4B12  CB3A      24          SRL  D
4B14  CB1B      25          RR   E
4B16  CB43      26          BIT  0,E                ;test abscisse paire
4B18  F5        27          PUSH AF
4B19  CD1DBC    28          CALL #BC1D              ;-> adresse ecran
4B1C  F1        29          POP  AF
4B1D  D1        30          POP  DE
4B1E  C1        31          POP  BC
4B1F  2012      32          JR   NZ,BCL3            ;affichage objet decale
                33 ;                                  si abscisse impaire
4B21  C5        34 BCL1:    PUSH BC                 ;affichage si abscisse
4B22  E5        35          PUSH HL                 ;paire
                36 ;
4B23  1A        37 BCL2:    LD   A,(DE)
4B24  AE        38          XOR  (HL)
4B25  77        39          LD   (HL),A             ;(HL)=(HL) XOR (DE)
4B26  23        40          INC  HL
```

```
4B27  13        41          INC  DE
4B28  10F9      42          DJNZ BCL2           ;test fin ligne
                43 ;
4B2A  E1        44          POP  HL
4B2B  CD26BC    45          CALL #BC26          ;ligne ecran suivante
4B2E  C1        46          POP  BC
4B2F  0D        47          DEC  C
4B30  20EF      48          JR   NZ,BCL1        ;test fin objet
4B32  C9        49          RET
                50 ;
4B33  C5        51 BCL3:    PUSH BC             ;affichage si abscisse
4B34  E5        52          PUSH HL             ;impaire
4B35  4E        53          LD   C,(HL)
                54 ;
4B36  1A        55 BCL4:    LD   A,(DE)
4B37  E6AA      56          AND  %10101010      ;point gauche
4B39  1F        57          RRA                 ;devient point droite
4B3A  A9        58          XOR  C              ;XOR avec octet ecran
4B3B  77        59          LD   (HL),A
4B3C  23        60          INC  HL             ;octet ecran suivant
4B3D  1A        61          LD   A,(DE)
4B3E  E655      62          AND  %01010101      ;point droite
4B40  17        63          RLA                 ;devient point gauche
4B41  AE        64          XOR  (HL)           ;XOR avec octet ecran
4B42  4F        65          LD   C,A            ;-> dans C
4B43  13        66          INC  DE             ;octet objet suivant
4B44  10F0      67          DJNZ BCL4           ;test fin ligne
                68 ;
4B46  E1        69          POP  HL
4B47  CD26BC    70          CALL #BC26          ;ligne ecran suivante
4B4A  C1        71          POP  BC
4B4B  0D        72          DEC  C
4B4C  20E5      73          JR   NZ,BCL3        ;test fin objet
4B4E  C9        74          RET
```

L'affichage et l'effacement de l'objet (méthode du tracé)

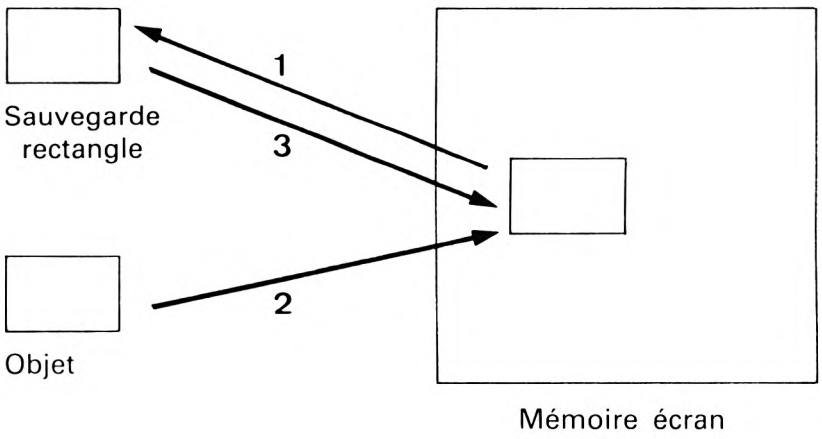
Les deux programmes que nous présentons affichent l'objet à une abscisse paire.

Le premier programme **TRACE1.ASS** échange la zone mémoire objet et le rectangle écran correspondant. Les avantages de cette méthode sont les suivants :

- la zone mémoire utilisée est minimale ;
- deux affichages consécutifs restituent l'image initiale ;
- rapidité de traitement.

Cependant, le rectangle objet étant échangé intégralement, si l'objet ne circonscrit pas le rectangle, il apparaîtra "au milieu" d'un rectangle vide à l'écran (on a perdu une partie des informations du fond) : le second programme **TRACE2.ASS** apporte une solution à cette anomalie : dans le cas général où l'objet ne remplit pas entièrement son rectangle, on ne doit afficher que la partie non nulle du rectangle. On doit donc obligatoirement sauvegarder le rectangle écran avant l'affichage de la partie non nulle du rectangle objet.

Le principe général peut être ainsi schématisé :



1 = *Sauvegarde de la zone de réception de l'objet.*
3 = *Restitution (réaffichage) du rectangle sauvegardé (après détection d'un déplacement).*
2 = *Affichage objet (partie non nulle du rectangle objet).*

Le programme TRACE2.ASS apporte donc la meilleure réponse au problème du déplacement de l'objet ; il exige cependant une zone mémoire plus importante, le traitement est plus lent, et il nous faut construire deux modules distincts (un pour l'affichage, l'autre pour l'effacement).

Voici les listes des **programmes TRACE1.ASS et TRACE2.ASS** :

```
1 ; --------  ECHANGE OBJET-ECRAN  --------
2 ;
3 ;-> Echange zone memoire objet et zone memoire ecran
4 ;
5 ;Entree: HL=adresse de l'objet
6 ;
```

```
4B00             7          ORG  19200
4B00  4E         8          LD   C,(HL)          ;hauteur objet
4B01  23         9          INC  HL
4B02  46        10          LD   B,(HL)          ;largeur objet
4B03  23        11          INC  HL
4B04  EB        12          EX   DE,HL           ;DE=debut dessin objet
4B05  CD1C4B    13          CALL ADR             ;HL=adresse ecran
                14 ;
4B08  C5        15 BCL1:    PUSH BC
4B09  E5        16          PUSH HL
                17 ;
4B0A  1A        18 BCL2:    LD   A,(DE)
4B0B  4E        19          LD   C,(HL)
4B0C  77        20          LD   (HL),A          ;echange objet-ecran
4B0D  79        21          LD   A,C
4B0E  12        22          LD   (DE),A
4B0F  23        23          INC  HL
4B10  13        24          INC  DE
4B11  10F7      25          DJNZ BCL2            ;test fin ligne
                26 ;
4B13  E1        27          POP  HL
4B14  CD26BC    28          CALL #BC26           ;ligne ecran suivante
4B17  C1        29          POP  BC
4B18  0D        30          DEC  C
4B19  20ED      31          JR   NZ,BCL1         ;test fin objet
4B1B  C9        32          RET
                33 ;
                34 ; Conversion coordonnees curs. graphique -> adresse ecran
                35 ;
4B1C  F5        36 ADR:     PUSH AF
4B1D  C5        37          PUSH BC
4B1E  D5        38          PUSH DE
4B1F  CDC6BB    39          CALL #BBC6           ;HL,DE=coord. curseur
                40 ;
4B22  CB3C      41          SRL  H               ;Y / 2
4B24  CB1D      42          RR   L
                43 ;
4B26  CB3A      44          SRL  D               ;X / 4
4B28  CB1B      45          RR   E
4B2A  CB3A      46          SRL  D
4B2C  CB1B      47          RR   E
                48 ;
4B2E  CD1DBC    49          CALL #BC1D           ;-> adresse ecran
4B31  D1        50          POP  DE
4B32  C1        51          POP  BC
4B33  F1        52          POP  AF
4B34  C9        53          RET
```

```
                    1 ; ------------  AFFICHAGE OBJET  ------------
                    2 ;
                    3 ;-> Sauvegarde rectangle ecran
                    4 ;   et ecrasement ecran par zone memoire objet non nulle
                    5 ;
                    6 ;Entree: HL=adresse de l'objet
                    7 ;
4B00                8         ORG  19200
4D58                9 SAUV:   EQU  19800
4B00  E5           10         PUSH HL
4B01  46           11         LD   B,(HL)          ;hauteur objet
4B02  23           12         INC  HL
4B03  4E           13         LD   C,(HL)          ;largeur objet
4B04  CD5E4B       14         CALL ADR             ;HL=adresse ecran
4B07  E5           15         PUSH HL
4B08  11584D       16         LD   DE,SAUV         ;adresse sauvegarde
                   17 ;                             du rectangle ecran
4B0B  C5           18 BCL1:   PUSH BC              ;ecran -> sauvegarde
4B0C  E5           19         PUSH HL
4B0D  0600         20         LD   B,0             ;BC=largeur
4B0F  EDB0         21         LDIR                 ;transfert ligne
4B11  E1           22         POP  HL
4B12  CD26BC       23         CALL #BC26           ;ligne ecran suivante
4B15  C1           24         POP  BC
4B16  10F3         25         DJNZ BCL1            ;test fin rectangle
4B18  D1           26         POP  DE              ;adresse ecran
4B19  E1           27         POP  HL              ;adresse objet
4B1A  4E           28         LD   C,(HL)          ;hauteur objet
4B1B  23           29         INC  HL
4B1C  46           30         LD   B,(HL)          ;largeur objet
4B1D  23           31         INC  HL
4B1E  EB           32         EX   DE,HL
                   33 ;
4B1F  C5           34 BCL2:   PUSH BC
4B20  E5           35         PUSH HL
4B21  1A           36 BCL3:   LD   A,(DE)
4B22  E6AA         37         AND  %10101010       ;pt gauche objet
4B24  2806         38         JR   Z,SUIT1         ;nul ?
4B26  4F           39         LD   C,A             ;non:
4B27  7E           40         LD   A,(HL)          ;octet ecran
4B28  E655         41         AND  %01010101       ;pt droite ecran
4B2A  B1           42         OR   C               ;+ pt gauche objet
4B2B  77           43         LD   (HL),A          ;dans ecran
4B2C  1A           44 SUIT1:  LD   A,(DE)
4B2D  E655         45         AND  %01010101       ;pt droite objet
4B2F  2806         46         JR   Z,SUIT2         ;nul ?
4B31  4F           47         LD   C,A             ;non :
4B32  7E           48         LD   A,(HL)          ;octet ecran
4B33  E6AA         49         AND  %10101010       ;pt gauche ecran
```

```
4B35  B1        50        OR   C                ;+ pt droite objet
4B36  77        51        LD   (HL),A           ;dans ecran
4B37  23        52 SUIT2: INC  HL
4B38  13        53        INC  DE
4B39  10E6      54        DJNZ BCL3             ;test fin ligne
                55 ;
                56 ;
4B3B  E1        57        POP  HL
4B3C  CD26BC    58        CALL #BC26            ;ligne ecran suivante
4B3F  C1        59        POP  BC
4B40  0D        60        DEC  C
4B41  20DC      61        JR   NZ,BCL2          ;test fin objet
4B43  C9        62        RET
                63 ;
                64 ; ---- EFFACEMENT OBJET  ----
                65 ;
                66 ;-> Restitution sauvegarde rectangle ecran
                67 ;
4B44  46        68        LD   B,(HL)           ;hauteur rectangle
4B45  23        69        INC  HL
4B46  4E        70        LD   C,(HL)           ;largeur rectangle
4B47  CD5E4B    71        CALL ADR              ;adresse ecran
4B4A  11584D    72        LD   DE,SAUV          ;adresse sauvegarde
4B4D  EB        73        EX   DE,HL
                74 ;
4B4E  C5        75 BCL4:  PUSH BC
4B4F  D5        76        PUSH DE
4B50  0600      77        LD   B,0              ;BC=largeur
4B52  EDB0      78        LDIR                  ;transfert
4B54  D1        79        POP  DE
4B55  EB        80        EX   DE,HL
4B56  CD26BC    81        CALL #BC26            ;ligne ecran suivante
4B59  EB        82        EX   DE,HL
4B5A  C1        83        POP  BC
4B5B  10F1      84        DJNZ BCL4             ;test derniere ligne
4B5D  C9        85        RET
                86 ;
                87 ; Conversion coordonnees curs. graphique -> adresse ecran
                88 ;
4B5E  F5        89 ADR:   PUSH AF
4B5F  C5        90        PUSH BC
4B60  D5        91        PUSH DE
4B61  CDC6BB    92        CALL #BBC6            ;HL,DE=coord. curseur
                93 ;
4B64  CB3C      94        SRL  H                ;Y / 2
4B66  CB1D      95        RR   L
                96 ;
4B68  CB3A      97        SRL  D                ;X / 4
4B6A  CB1B      98        RR   E
```

```
4B6C  CB3A      99        SRL  D
4B6E  CB1B     100        RR   E
               101 ;
4B70  CD1DBC   102        CALL #BC1D              ;-> adresse ecran
4B73  D1       103        POP  DE
4B74  C1       104        POP  BC
4B75  F1       105        POP  AF
4B76  C9       106        RET
```

LA GESTION DU DÉPLACEMENT D'UN CARACTÈRE

Des matrices de caractères sont inscrites dans la ROM de l'Amstrad, mais l'utilisateur peut construire lui-même ses matrices ; une matrice de caractère permet de construire un "objet-caractère" inscrit dans un rectangle de 8 lignes de 8 points dans le mode "0" (8 octets codent l'objet, le bit 7 d'un octet correspond au point le plus à gauche) ; il est possible de gérer des tables de matrices de 256 éléments.

La routine $BBFC réalise l'affichage d'une matrice de points (caractère ou objet) à la position courante du curseur graphique ; la position graphique courante correspond au point supérieur gauche de la matrice ; après l'affichage, cette position est déplacée d'un caractère (soit 32 points en mode "0", ou 8 en mode "2") vers la droite.

La routine $BC59 active le mode d'écriture graphique (mode "nouvelle encre", mode XOR, AND ou OR) ; le choix du mode XOR permet d'effacer le caractère avant l'affichage à la nouvelle position.

L'avantage de cette méthode est la rapidité du déplacement ; en revanche, l'objet est de dimensions fixes assez réduites et une seule couleur est possible.

Voici la liste du programme Assembleur DEPLC.ASS (ce programme fait appel au module de validité de déplacement VALID.ASS) :

```
                1 ; --------  DEPLACEMENT D'UN CARACTERE  --------
                2 ;
                3 ;            PROGRAMME PRINCIPAL
                4 ;
4A38            5         ORG  19000
4A38  1801      6         JR   DEBUT
4A3A  E6        7 CAR:    DEFB 230
4A9C            8 VALID:  EQU  19100              ;s/p de deplacement
```

```
4A3B  3E01        9 DEBUT:  LD   A,1
4A3D  CD59BC     10         CALL #BC59          ;mode graph. XOR
4A40  CD6A4A     11         CALL AFFCAR         ;affichage caractere
                 12 ;
4A43  3E42       13 RELECT: LD   A,66          ;code touche ESC
4A45  CD1EBB     14         CALL #BB1E          ;test touche
4A48  C0         15         RET  NZ             ;retour si appuyee
4A49  CD24BB     16         CALL #BB24          ;lecture joystick
4A4C  E60F       17         AND  #0F
4A4E  28F3       18         JR   Z,RELECT       ;relecture si rien
                 19 ;
4A50  CD9C4A     20         CALL VALID          ;deplacement
4A53  38EE       21         JR   C,RELECT       ;relecture si deplacement
                 22 ;                            impossible
4A55  E5         23         PUSH HL             ;sauv. nouvelle position
4A56  D5         24         PUSH DE
4A57  CD19BD     25         CALL #BD19          ;synchronisation
4A5A  CD6A4A     26         CALL AFFCAR         ;effacement caractere
4A5D  D1         27         POP  DE             ;rest. nouvelle position
4A5E  E1         28         POP  HL
4A5F  CDC0BB     29         CALL #BBC0          ;mise a jour pos. curseur
4A62  CD19BD     30         CALL #BD19          ;synchronisation
4A65  CD6A4A     31         CALL AFFCAR         ;affichage caractere
4A68  18D9       32         JR   RELECT
                 33 ;
4A6A  3A3A4A     34 AFFCAR: LD   A,(CAR)        ;n0 caractere
4A6D  CDFCBB     35         CALL #BBFC          ;affichage
4A70  CDC6BB     36         CALL #BBC6          ;demande pos. curseur
4A73  01E0FF     37         LD   BC,-32         ;restauration
4A76  EB         38         EX   DE,HL          ;abscisse curseur
4A77  09         39         ADD  HL,BC
4A78  EB         40         EX   DE,HL
4A79  CDC0BB     41         CALL #BBC0          ;positionne curseur
4A7C  C9         42         RET
```

LA VITESSE DE DÉPLACEMENT ET LA SYNCHRONISATION D'AFFICHAGE

Nous avons trois méthodes à notre disposition pour ajuster la vitesse de déplacement d'un objet.

La modification des pas X et Y du déplacement

Le réglage de la vitesse est imprécis, mais efficace ; si l'on désire visualiser un déplacement rapide, on choisira une valeur de pas assez

grande ; en revanche, le déplacement d'un crayon et la visualisation du tracé associé exige un pas fin si l'on veut dessiner avec précision.

La boucle de temporisation

Une boucle de temporisation permet d'effectuer un réglage fin ; la valeur de la variable de temporisation (nombre de passages dans la boucle), et la connaissance du nombre de cycles des instructions utilisées permettent de calculer précisément le temps de temporisation.

Voici un exemple :

```
        LD HL,100 → valeur temporisation

BA :    DEC HL
        LD A,H
        OR L
        JR NZ,BA
```

Possibilité de synchroniser l'affichage de l'objet avec l'affichage de l'écran

On a pu remarquer, à la lecture des programmes précédents, l'utilisation de la routine $BD19 ; l'appel de cette routine provoque une attente de début de balayage vertical (l'interruption est générée par le circuit CRT6845, chargé de gérer l'affichage sur le moniteur vidéo).

On a ainsi une image stable au cours du déplacement de l'objet (pas d'effet de clignotement).

En contrepartie, cette synchronisation est coûteuse en temps de traitement ; en effet, l'interruption de début d'affichage est générée tous les 1/50^{e} de seconde ; par conséquent, si l'on fait un appel à cette routine avant l'affichage et l'effacement, le temps de déplacement vertical d'un objet sera (hypothèses : déplacement point par point, on néglige le temps d'affichage de l'objet) :

$$t = 2 * (1/50) * 200 = 8''$$

On peut donc programmer la synchronisation dans les cas où la lenteur qu'elle induit n'est pas pénalisante (dans les programmes proposés, on pourra remplacer l'appel CALL $BD19 par trois instructions NOP) ; si l'objet à afficher est de dimensions réduites, on pourra se contenter d'un seul appel à la synchronisation avant "affichage/effacement", ce qui divise le temps de déplacement par 2.

Chapitre 4

LES EXEMPLES PRATIQUES DE FONCTIONS DE DESSIN

L'OUTIL DE DESSIN

Les exemples que nous présentons fonctionnent avec un joystick ou une souris (AMX mouse par exemple). Le principe de fonctionnement de la souris est identique à celui du joystick : lorsqu'on agit sur la souris, celle-ci génère quatre signaux correspondants aux directions de déplacement ; le logiciel AMX CONTROL associé à la souris incrémente ou décrémente divers compteurs en fonction des signaux reçus ; les compteurs gèrent des coordonnées "x" et "y" compatibles avec le système de coordonnées graphiques (le couple de valeurs 0,0 correspond au point en bas et à gauche de l'écran). Les routines de AMX CONTROL permettent de lire les coordonnées.

Remarque sur l'exploitation en Basic des applications de dessin présentées ci-après : d'une manière générale, on programme les étapes suivantes :

- chargement du programme principal (dessin, gomme, aérostaphe...) à l'adresse définie par la commande Assembleur ORG (figurant en tête de chaque programme) ;
- chargement des différents sous-programmes appelés (VALID.ASS/ XOR2.ASS) ;
- exécution d'un CALL au point d'entrée défini par l'instruction ORG du programme principal ;

soit par exemple :

```
 5 MODE Ø : MEMORY 15ØØØ
10 LOAD "DEPLF.ASS". 19ØØØ
20 LOAD "VALID.ASS", 191ØØ
30 LOAD "XOR.ASS". 192ØØ
40 ← CALL 19ØØØ
50 END
```

LES APPLICATIONS DE DESSIN INTERACTIVES

Le tracé libre avec le joystick ou la souris

Le principe général du tracé est simple à mettre en œuvre ; la difficulté est de réaliser un "outil" à la fois sensible et précis. Si vous arrivez à écrire votre prénom lisiblement et d'un seul jet, alors le pari est gagné.

La routine $BB24 déjà utilisée dans les programmes du chapitre,&14 permet de lire l'état du joystick ; les bits de l'octet d'état positionnés à "1" indiquent l'action détectée :

- bit 0 : déplacement vers le haut ;
- bit 1 : déplacement vers le bas ;
- bit 2 : déplacement à gauche ;
- bit 3 : déplacement à droite ;
- bit 4 : bouton 2 appuyé ou ;
- bit 5 : bouton 1 appuyé ;
- bit 6 : (réservé) ;
- bit 7 : toujours à 0.

La mise à jour des nouvelles coordonnées est fournie par l'appel de la routine $BBCO (en entrée, le registre DE contient l'abscisse "x", et HL l'ordonnée "y"). Le bit 4 est testé afin de détecter le mode de travail (tracé ou déplacement sans tracé).

Un tracé continu est obtenu en traçant une droite entre la nouvelle position et l'ancienne : la routine $BBF6 trace une droite depuis la position précédente, vers la nouvelle position (en entrée, les registres DE et HL contiennent les coordonnées usager du point d'arrivée).

Le programme TRACE.ASS utilise la routine de déplacement (test de la validité du déplacement) et une des routines d'affichage présentée dans le &14 ; cette routine affiche l'objet symbolisant l'outil de dessin (un crayon).

Voici la liste du **programme TRACE.ASS :**

```
                    1 ; -------- TRACE : PROGRAMME PRINCIPAL ----------
                    2 ;
4A38                3         ORG  19000
4A9C                4 VALID:  EQU  19100          ;s/p deplacement
4B00                5 AFFOBJ: EQU  19200          ;s/p affichage
4A38  1804          6         JR   DEBUT
4A3A  01            7 OBJ:    DEFB 1              ;n0 objet
4A3B  00            8 DRAP:   DEFB 0              ;drapeau mode trace
4A3C  9156          9 ADROBJ: DEFW 22161          ;adresse objet
4A3E  3A3A4A       10 DEBUT:  LD   A,(OBJ)        ;n0 de l'objet
4A41  21C755       11         LD   HL,21959       ;adr 1er objet - 202
4A44  11CA00       12         LD   DE,202         ;longueur objet
4A47  47           13         LD   B,A
4A48  19           14 SUIVAN: ADD  HL,DE          ;HL=adresse objet
4A49  10FD         15         DJNZ SUIVAN
4A4B  223C4A       16         LD   (ADROBJ),HL    ;stockage adresse
4A4E  CD004B       17         CALL AFFOBJ         ;affichage objet
4A51  3E42         18 RELECT: LD   A,66            ;code touche ESC
4A53  CD1EBB       19         CALL #BB1E          ;test touche
4A56  C0           20         RET  NZ             ;retour si appuyee
4A57  CD24BB       21         CALL #BB24          ;lecture joystick
4A5A  CB67         22         BIT  4,A            ;feu ?
4A5C  280E         23         JR   Z,MASC         ;MASC si non
4A5E  3A3B4A       24         LD   A,(DRAP)       ;activation ou
4A61  2F           25         CPL                 ;desactivation
4A62  323B4A       26         LD   (DRAP),A       ;du mode trace
4A65  CD24BB       27 ENCORE: CALL #BB24
4A68  CB67         28         BIT  4,A            ;attend relachement
4A6A  20F9         29         JR   NZ,ENCORE      ;touche "FEU"
4A6C  A7           30 MASC:   AND  A
4A6D  28E2         31         JR   Z,RELECT       ;relecture si rien
4A6F  CD9C4A       32         CALL VALID
4A72  38DD         33         JR   C,RELECT       ;relecture si deplacement
                    34 ;                             impossible
4A74  E5           35         PUSH HL             ;sauv. nouvelle position
4A75  D5           36         PUSH DE
4A76  CD19BD       37         CALL #BD19          ;synchronisation
4A79  2A3C4A       38         LD   HL,(ADROBJ)
4A7C  CD004B       39         CALL AFFOBJ         ;effacement objet
4A7F  D1           40         POP  DE             ;rest. nouvelle position
4A80  E1           41         POP  HL
4A81  3A3B4A       42         LD   A,(DRAP)       ;teste si le mode
4A84  A7           43         AND  A              ;trace est active
4A85  2807         44         JR   Z,SUITE
4A87  E5           45         PUSH HL
4A88  D5           46         PUSH DE
```

```
4A89  CDF6BB      47          CALL #BBF6              ;trace une droite depuis
4A8C  D1          48          POP  DE                 ;la position du curseur
4A8D  E1          49          POP  HL
4A8E  CDC0BB      50 SUITE:   CALL #BBC0              ;mise a jour pos. curs
4A91  CD19BD      51          CALL #BD19              ;synchronisation
4A94  2A3C4A      52          LD   HL,(ADROBJ)
4A97  CD004B      53          CALL AFFOBJ             ;affichage objet
4A9A  18B5        54          JR   RELECT
```

La gomme

Complément indispensable du "crayon", le programme GOMME.ASS va nous permettre de gommer notre travail précédent.

Ce programme est construit comme précédemment ; la valeur de la couleur d'encre est donnée par appel de la routine $BBDE ("0" = noir) ; le tracé des lignes correspond donc à un tracé en noir (points éteints).

Voici la liste du **programme GOMME.ASS** :

```
                    1 ; -------- GOMME : PROGRAMME PRINCIPAL ---------
                    2 ;
4A33                3          ORG  18995
4A9C                4 VALID:   EQU  19100              ;s/p de deplacement
4B00                5 AFFOBJ:  EQU  19200              ;s/p d'affichage
4A33  1804          6          JR   DEBUT
4A35  02            7 OBJ:     DEFB 2                  ;n0 de l'objet
4A36  00            8 DRAP:    DEFB 0
4A37  9156          9 ADROBJ:  DEFW 22161
4A39  3A354A       10 DEBUT:   LD   A,(OBJ)            ;n0 de l'objet
4A3C  21C755       11          LD   HL,21959           ;adr 1er objet - 202
4A3F  11CA00       12          LD   DE,202             ;longueur objet
4A42  47           13          LD   B,A
4A43  19           14 SUIVAN:  ADD  HL,DE              ;HL=adresse objet
4A44  10FD         15          DJNZ SUIVAN
4A46  22374A       16          LD   (ADROBJ),HL        ;stockage adresse
4A49  CD004B       17          CALL AFFOBJ             ;affichage objet
4A4C  3E00         18          LD   A,0                ;couleur du fond
4A4E  CDDEBB       19          CALL #BBDE              ;->couleur graphiques
4A51  3E42         20 RELECT:  LD   A,66                ;code touche ESC
4A53  CD1EBB       21          CALL #BB1E              ;test touche
4A56  C0           22          RET  NZ                 ;retour si appuyee
4A57  CD24BB       23          CALL #BB24              ;lecture joystick
4A5A  CB67         24          BIT  4,A                ;feu?
4A5C  280E         25          JR   Z,MASC             ;saut si non
```

```
4A5E  3A364A      26          LD   A,(DRAP)
4A61  2F          27          CPL                    ;inversion drapeau
4A62  32364A      28          LD   (DRAP),A
4A65  CD24BB      29 ENCORE:  CALL #BB24             ;attente relachement
4A68  CB67        30          BIT  4,A               ;de la touche "FEU"
4A6A  20F9        31          JR   NZ,ENCORE
4A6C  A7          32 MASC:    AND  A
4A6D  28E2        33          JR   Z,RELECT          ;relecture si rien
4A6F  CD9C4A      34          CALL VALID             ;deplacement
4A72  38DD        35          JR   C,RELECT          ;relecture si deplacement
4A74  E5          36          PUSH HL                ;sauv. nouvelle position
4A75  D5          37          PUSH DE
4A76  CD19BD      38          CALL #BD19             ;synchronisation
4A79  2A374A      39          LD   HL,(ADROBJ)
4A7C  CD004B      40          CALL AFFOBJ            ;effacement objet
4A7F  D1          41          POP  DE                ;rest. nouvelle position
4A80  E1          42          POP  HL
4A81  3A364A      43          LD   A,(DRAP)
4A84  A7          44          AND  A                 ;mode trace?
4A85  2807        45          JR   Z,SUITE           ;saut si non
4A87  E5          46          PUSH HL
4A88  D5          47          PUSH DE
4A89  CDF6BB      48          CALL #BBF6             ;trace une droite depuis
4A8C  D1          49          POP  DE                ;la position du curseur
4A8D  E1          50          POP  HL
4A8E  CDC0BB      51 SUITE:   CALL #BBC0             ;mise a jour pos. curs
4A91  CD19BD      52          CALL #BD19             ;synchronisation
4A94  2A374A      53          LD   HL,(ADROBJ)
4A97  CD004B      54          CALL AFFOBJ            ;affichage objet
4A9A  18B5        55          JR   RELECT
```

L'aérographe

L'aérographe est un "pistolet à peinture" fonctionnant de la façon suivante : l'action du joystick permet de déplacer à l'écran une petite fenêtre de repérage ; lorsqu'on est en mode "tracé" (il faut cliquer pour activer ou sortir du mode "tracé"), des points colorés sont inscrits aléatoirement à l'intérieur de l'objet ; c'est donc le temps de passage sur une zone donnée et l'amplitude des mouvements effectués avec le joystick qui conditionnent l'intensité de la "vaporisation".

Le programme AERO.ASS comprend quatre modules :

- une boucle principale de gestion des commandes ;
- un sous-programme de déplacement ;
- un sous-programme d'affichage de l'objet de repérage ;
- un sous-programme d'affichage aléatoire de points.

Les paramètres agissant sur la concentration des points émis sont les suivants :

- le compteur CPT détermine l'intervalle d'affichage des points ;
- le nombre de points affichés par étape ;
- la mise à jour de CPT en cas de déplacement.

La méthode de calcul de la position aléatoire d'un point est la suivante : la position initiale du curseur graphique correspond au point en haut et à gauche de l'objet de repérage ; à partir de cette position, on calcule une colonne et une ligne aléatoires (valeurs comprises entre 0 et 7) et on affiche le point correspondant.

La formule Xi = (Xi−1 * 9) + 1 permet de produire un nombre aléatoire compris entre 0 et 255.

SEED = (SEED * 9) + 1 modulo 256
et A = SEED modulo 8

On obtient huit nombres différents, puis la séquence aléatoire se répète à nouveau ; cette "production" n'est pas très utilisable pour notre aérographe ; nous avons utilisé la variante suivante :

SEED = (SEED + R)* + 1 modulo 256
et A = SEED modulo 8

Le registre R est utilisé pour le rafraîchissement des mémoires, et varie constamment ; cette méthode est donc satisfaisante.

Voici la liste du **programme AERO.ASS** (le cadre de repérage est défini par la commande Basic SYMBOL dans le programme de menu de la face "A" ; la validité de son déplacement étant testée dans le module VALID.ASS) :

```
                    1 ; -------- AEROGRAPHE --------
                    2 ;
                    3 ;     PROGRAMME PRINCIPAL
                    4 ;
4A38                5         ORG  19000
4A38  180C          6         JR   DEBUT
4A3A  0400          7 PASX:   DEFW 4                  ;pas X
4A3C  0200          8 PASY:   DEFW 2                  ;pas Y
4A3E  4002          9 LIMX:   DEFW 576                ;limite superieure X
4A40  2000         10 LIMY:   DEFW 32                 ;limite inferieure Y
4A42  FD           11 CAR:    DEFB 253                ;curseur
4A43  00           12 DRAP:   DEFB 0                  ;mode trace
4A44  14           13 CPT:    DEFB 20                 ;compteur
```

```
4A45  03        14 SEED:   DEFB 3                ;base pour nbre aleatoire
4A46  3E01      15 DEBUT:  LD   A,1
4A48  CD59BC    16         CALL #BC59            ;mode graph. XOR
4A4B  CDE04A    17         CALL AFFCAR           ;affichage caractere
4A4E  21FF0F    18 BT:     LD   HL,#FFF          ;temporisation en cas
4A51  2B        19 BA:     DEC  HL               ;de non deplacement
4A52  7C        20         LD   A,H
4A53  B5        21         OR   L
4A54  20FB      22         JR   NZ,BA
4A56  3A434A    23 RELECT: LD   A,(DRAP)         ;affichage points
4A59  A7        24         AND  A                ;si mode trace active
4A5A  C4F34A    25         CALL NZ,POINT
4A5D  3E42      26         LD   A,66             ;code touche ESC
4A5F  CD1EBB    27         CALL #BB1E            ;test touche
4A62  C0        28         RET  NZ               ;retour si appuyee
4A63  CD24BB    29         CALL #BB24            ;lecture joystick
4A66  CB67      30         BIT  4,A              ;test touche "FEU"
4A68  280E      31         JR   Z,SUITE
4A6A  3A434A    32         LD   A,(DRAP)
4A6D  2F        33         CPL                   ;inversion du mode trace
4A6E  32434A    34         LD   (DRAP),A
4A71  CD24BB    35 ENCORE: CALL #BB24            ;attente relachement
4A74  CB67      36         BIT  4,A              ;de la touche "FEU"
4A76  20F9      37         JR   NZ,ENCORE
4A78  A7        38 SUITE:  AND  A
4A79  28D3      39         JR   Z,BT             ;relecture si rien
                40 ;
4A7B  CD944A    41         CALL VALID            ;deplacement
4A7E  38CE      42         JR   C,BT             ;relecture si deplacement
                43 ;                               impossible
4A80  E5        44         PUSH HL               ;sauv. nouvelle position
4A81  D5        45         PUSH DE
4A82  CDE04A    46         CALL AFFCAR           ;effacement caractere
4A85  D1        47         POP  DE               ;rest. nouvelle position
4A86  E1        48         POP  HL
4A87  CDC0BB    49         CALL #BBC0            ;mise a jour pos. curseur
4A8A  CDE04A    50         CALL AFFCAR           ;affichage caractere
4A8D  3E01      51         LD   A,1              ;preparation de
4A8F  32444A    52         LD   (CPT),A          ;l'affichage de points
4A92  18C2      53         JR   RELECT
                54
                55 ; --------  TEST VALIDITE DEPLACEMENT  --------
                56 ;
                57 ;Entree: A=etat joystick
                58 ;
                59 ;Sortie: si Cy=1 deplacement impossible
                60 ;        si Cy=0 deplacement possible
                61 ;                et HL,DE = nouvelles coordonnes
                62 ;
```

```
4A94  F5          63 VALID:  PUSH AF
4A95  CDC6BB      64         CALL #BBC6              ;HL,DE=coordonnees curs.
4A98  F1          65         POP  AF
4A99  1F          66 HAUT:   RRA                     ;Cy=etat joystick haut
4A9A  300F        67         JR   NC,BAS
4A9C  ED4B3C4A    68         LD   BC,(PASY)
4AA0  09          69         ADD  HL,BC              ;nouvelle coord. Y
4AA1  E5          70         PUSH HL
4AA2  010001      71         LD   BC,400             ;limite sup Y
4AA5  B7          72         OR   A
4AA6  ED42        73         SBC  HL,BC              ;Y < limite sup ?
4AA8  E1          74         POP  HL
4AA9  3F          75         CCF
4AAA  D8          76         RET  C                  ;retour si non
4AAB  1F          77 BAS:    RRA                     ;Cy=etat joystick bas
4AAC  3011        78         JR   NC,GAUCHE
4AAE  ED4B3C4A    79         LD   BC,(PASY)
4AB2  B7          80         OR   A
4AB3  ED42        81         SBC  HL,BC              ;nouvelle coord. Y
4AB5  D8          82         RET  C                  ;retour si Y < 0
4AB6  E5          83         PUSH HL
4AB7  ED4B404A    84         LD   BC,(LIMY)
4ABB  ED42        85         SBC  HL,BC              ;Y > limite inf ?
4ABD  E1          86         POP  HL
4ABE  D8          87         RET  C                  ;retour si non
4ABF  EB          88 GAUCHE: EX   DE,HL
4AC0  1F          89         RRA                     ;Cy=etat joystick gauche
4AC1  3008        90         JR   NC,DROITE
4AC3  ED4B3A4A    91         LD   BC,(PASX)
4AC7  B7          92         OR   A
4AC8  ED42        93         SBC  HL,BC              ;nouvelle coordonnee X
4ACA  D8          94         RET  C                  ;retour si X < 0
4ACB  1F          95 DROITE: RRA                     ;Cy=etat joystick droite
4ACC  3010        96         JR   NC,FIN
4ACE  ED4B3A4A    97         LD   BC,(PASX)
4AD2  09          98         ADD  HL,BC              ;nouvelle coordonnee X
4AD3  E5          99         PUSH HL
4AD4  ED4B3E4A   100         LD   BC,(LIMX)
4AD8  B7         101         OR   A
4AD9  ED42       102         SBC  HL,BC              ;X < limite sup ?
4ADB  E1         103         POP  HL
4ADC  3F         104         CCF
4ADD  D8         105         RET  C                  ;retour si non
4ADE  EB         106 FIN:    EX   DE,HL
4ADF  C9         107         RET
                 108
                 109 ; -----  AFFICHAGE CARACTERE  -----
                 110
4AE0  3A424A     111 AFFCAR: LD   A,(CAR)             ;n0 caractere
```

```
4AE3  CDFCBB    112         CALL #BBFC              ;affichage
4AE6  CDC6BB    113         CALL #BBC6              ;demande pos. curseur
4AE9  01E0FF    114         LD   BC,-32             ;restauration
4AEC  EB        115         EX   DE,HL              ;abscisse curseur
4AED  09        116         ADD  HL,BC
4AEE  EB        117         EX   DE,HL
4AEF  CDC0BB    118         CALL #BBC0              ;positionne curseur
4AF2  C9        119         RET
                120
                121 ; ---- AFFICHAGE POINTS ALEATOIRES ----
                122
4AF3  21444A    123 POINT:  LD   HL,CPT             ;compteur-1
4AF6  35        124         DEC  (HL)
4AF7  C0        125         RET  NZ                 ;retour si <>0
4AF8  3614      126         LD   (HL),20            ;recharge compteur
4AFA  CDE04A    127         CALL AFFCAR             ;effacement caractere
4AFD  AF        128         XOR  A
4AFE  CD59BC    129         CALL #BC59              ;mode graph. normal
4B01  CDC6BB    130         CALL #BBC6              ;position curseur
4B04  0602      131         LD   B,2                ;nbre de pts a ecrire
4B06  C5        132 BCLE:   PUSH BC
4B07  E5        133         PUSH HL
4B08  D5        134         PUSH DE
4B09  CD2D4B    135         CALL ALEA               ;ligne aleatoire
4B0C  0600      136         LD   B,0
4B0E  4F        137         LD   C,A
4B0F  ED42      138         SBC  HL,BC
4B11  CD2D4B    139         CALL ALEA               ;colonne aleatoire
4B14  17        140         RLA                     ; * 2
4B15  83        141         ADD  A,E
4B16  5F        142         LD   E,A
4B17  3001      143         JR   NC,NC2
4B19  14        144         INC  D
4B1A  CDEABB    145 NC2:    CALL #BBEA              ;affiche un point
4B1D  D1        146         POP  DE
4B1E  E1        147         POP  HL
4B1F  C1        148         POP  BC
4B20  10E4      149         DJNZ BCLE               ;point suivant
4B22  CDC0BB    150         CALL #BBC0              ;restauration coord.curs
4B25  3E01      151         LD   A,1
4B27  CD59BC    152         CALL #BC59              ;mode graph. XOR
4B2A  C3E04A    153         JP   AFFCAR             ;affichage car. + retour
                154 ;
                155 ;---- NOMBRE ALEATOIRE ----
                156 ;
                157 ;en sortie A=entier pair entre 0 et 14
                158 ;
4B2D  ED5F      159 ALEA:   LD   A,R
4B2F  4F        160         LD   C,A
```

```
4B30  3A454A      161       LD   A,(SEED)          ;nbre de base
4B33  81          162       ADD  A,C               ; + R
4B34  4F          163       LD   C,A
4B35  CB27        164       SLA  A                 ; * 9
4B37  CB27        165       SLA  A
4B39  81          166       ADD  A,C
4B3A  3C          167       INC  A                 ; + 1
4B3B  32454A      168       LD   (SEED),A          ;nbre de base suiv.
4B3E  E607        169       AND  7                 ;0 <= A <= 7
4B40  17          170       RLA                    ; * 2
4B41  C9          171       RET
```

LES TRANSFORMATIONS D'IMAGES

Les programmes présentés dans ce chapitre permettent de transformer des portions d'images (agrandissement, inversion de couleurs et symétries). Les portions d'images affectées par les transformations sont choisies de manière interactive par le déplacement d'un cadre ou fenêtre de travail.

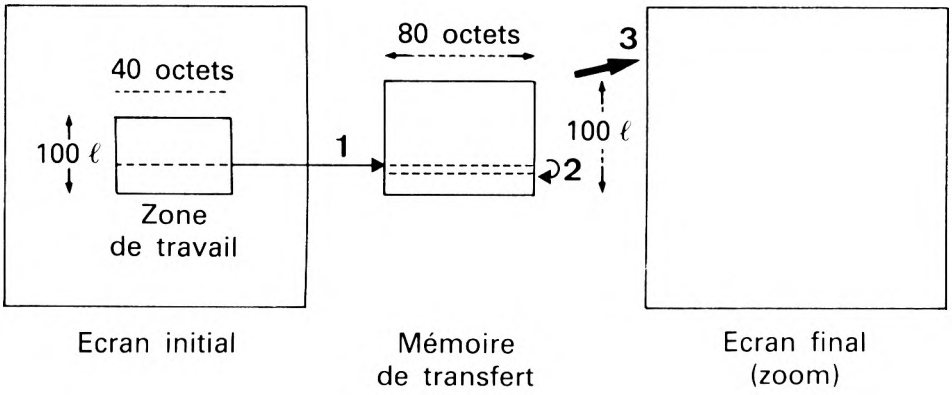
Le zoom "couleur" (facteur d'agrandissement = 2 dans les deux directions)

Nous allons agrandir l'image contenue dans une fenêtre (forme rectangulaire) que nous pouvons déplacer à l'aide du joystick sur une image ; dès que l'on clique, le programme zoom calcule et affiche la zone agrandie ; on peut à nouveau choisir une zone à agrandir sur cette nouvelle image.

Nous utilisons les trois programmes suivants :

- le programme **DEPLZ.ASS** est la boucle principale de gestion des commandes ; il fait appel au sous-programme de déplacement ;
- le programme **AFFCAD.ASS** affiche la fenêtre de travail ;
- le programme **ZOOM.ASS** calcule la nouvelle image.

Le principe de fonctionnement du programme ZOOM.ASS peut être schématisé de la façon suivante :



1 = *Dédoublement des points sur la ligne.*
2 = *Recopie d'une ligne.*
3 = *Affichage final.*

Voici les listes des **trois programmes** :

```
                    1 ; --------          ZOOM              --------
                    2 ;
                    3 ;            PROGRAMME PRINCIPAL
                    4 ;
4A38                5          ORG  19000
4A9C                6 VALID:   EQU  19100          ;s/p de deplacement
4B00                7 AFFCAD:  EQU  19200          ;s/p affichage cadre
4B64                8 ZOOM:    EQU  19300          ;s/p zoom du cadre
4A38  3E01          9          LD   A,1
4A3A  CD59BC       10          CALL #BC59          ;mode graph. XOR
4A3D  CD004B       11          CALL AFFCAD         ;affichage cadre
                   12 ;
4A40  3E42         13 RELECT: LD   A,66            ;code touche ESC
```

```
4A42  CD1EBB     14          CALL #BB1E              ;test touche
4A45  C0         15          RET  NZ                 ;retour si appuyee
4A46  CD24BB     16          CALL #BB24              ;lecture joystick
4A49  A7         17          AND  A
4A4A  28F4       18          JR   Z,RELECT           ;relecture si rien
4A4C  CB67       19          BIT  4,A
4A4E  280B       20          JR   Z,PACLIC           ;zoom si "clic"
                 21 ;
4A50  CD004B     22          CALL AFFCAD             ;effacement cadre
4A53  CD644B     23          CALL ZOOM               ;zoom
4A56  CD004B     24          CALL AFFCAD             ;affichage cadre
4A59  18E5       25          JR   RELECT
                 26 ;
4A5B  CD9C4A     27 PACLIC:  CALL VALID              ;deplacement
4A5E  38E0       28          JR   C,RELECT           ;relecture si deplacement
                 29 ;                                  impossible
4A60  E5         30          PUSH HL                 ;sauv. nouvelle position
4A61  D5         31          PUSH DE
4A62  CD004B     32          CALL AFFCAD             ;effacement cadre
4A65  D1         33          POP  DE                 ;rest. nouvelle position
4A66  E1         34          POP  HL
4A67  CDC0BB     35          CALL #BBC0              ;mise a jour pos. curseur
4A6A  CD004B     36          CALL AFFCAD             ;affichage cadre
4A6D  18D1       37          JR   RELECT
```

```
                  1 ; -------- AFFICHAGE CADRE ZOOM --------
                  2 ;
                  3 ; affichage d'un cadre dont le coin en haut
                  4 ; a gauche est la position du curseur graphique
                  5 ;
4B00              6           ORG  19200
4B00  210000      7           LD   HL,0
4B03  113C01      8           LD   DE,316             ;largeur cadre
4B06  CDF9BB      9           CALL #BBF9              ;bord haut
4B09  213AFF     10           LD   HL,-198
4B0C  110000     11           LD   DE,0
4B0F  CDF9BB     12           CALL #BBF9              ;bord droite
4B12  210000     13           LD   HL,0
4B15  11C4FE     14           LD   DE,-316
4B18  CDF9BB     15           CALL #BBF9              ;bord bas
4B1B  21C600     16           LD   HL,198             ;hauteur cadre
4B1E  110000     17           LD   DE,0
4B21  CDF9BB     18           CALL #BBF9              ;bord gauche
4B24  C9         19           RET
```

```
                    1 ; ---------- ZOOM ----------
                    2 ;
                    3 ;grossissement x2 en largeur et
                    4 ;x2 en hauteur d'une fenetre de
                    5 ;l'ecran :
                    6 ;    - dont le point en haut a
                    7 ;      gauche est la position
                    8 ;      courante du curseur graph.
                    9 ;    - de largeur 40 octets,
                   10 ;    - de hauteur 100 lignes.
                   11 ;
                   12 ;zone memoire de transfert:
                   13 ;   - debut: 24900
                   14 ;   - longueur: 16384 octets
                   15 ;
4B64               16         ORG  19300
4B64  CDC6BB       17 ZOOM:   CALL #BBC6              ;HL,DE=coord. curseur
4B67  CDCD4B       18         CALL CONV               ;->adresse ecran
                   19 ;
                   20 ;grossissement horizontal x2
                   21 ;
4B6A  114461       22         LD   DE,24900           ;debut zone de transfert
4B6D  0619         23         LD   B,25               ;nbre de lignes / bloc
                   24 ;
4B6F  C5           25 BCL1:   PUSH BC
4B70  D5           26         PUSH DE
4B71  0604         27         LD   B,4                ;un bloc sur deux
                   28 ;
4B73  C5           29 BCL2:   PUSH BC
4B74  E5           30         PUSH HL
4B75  D5           31         PUSH DE
4B76  0628         32         LD   B,40               ;largeur cadre
                   33 ;
4B78  7E           34 BCL3:   LD   A,(HL)
4B79  E6AA         35         AND  %10101010          ;point gauche
4B7B  4F           36         LD   C,A
4B7C  1F           37         RRA
4B7D  B1           38         OR   C                  ;multiplication par 2
4B7E  12           39         LD   (DE),A
4B7F  13           40         INC  DE
4B80  7E           41         LD   A,(HL)
4B81  E655         42         AND  %01010101          ;point droite
4B83  4F           43         LD   C,A
4B84  17           44         RLA
4B85  B1           45         OR   C                  ;multiplication par 2
4B86  12           46         LD   (DE),A
4B87  13           47         INC  DE
4B88  23           48         INC  HL
4B89  10ED         49         DJNZ BCL3
                   50 ;
```

```
4B8B  D1        51        POP  DE
4B8C  210010    52        LD   HL,#1000
4B8F  19        53        ADD  HL,DE
4B90  EB        54        EX   DE,HL          ;DE=lg suiv. ds zone transfert
4B91  E1        55        POP  HL
4B92  CD26BC    56        CALL #BC26          ;ligne ecran suivante
4B95  C1        57        POP  BC
4B96  10DB      58        DJNZ BCL2
                59 ;
4B98  D1        60        POP  DE
4B99  EB        61        EX   DE,HL
4B9A  015000    62        LD   BC,80
4B9D  09        63        ADD  HL,BC
4B9E  EB        64        EX   DE,HL          ;DE=lg suiv. ds zone transfert
4B9F  C1        65        POP  BC
4BA0  10CD      66        DJNZ BCL1
                67 ;
                68 ;grossissement vertical x2
                69 ;
4BA2  114461    70        LD   DE,24900       ;debut zone de transfert
4BA5  0604      71        LD   B,4            ;nbre de blocs a recopier
4BA7  C5        72 BCL4:  PUSH BC
4BA8  210008    73        LD   HL,#800
4BAB  19        74        ADD  HL,DE
4BAC  EB        75        EX   DE,HL          ;HL=debut bloc
                76 ;                         DE=debut bloc suivant
4BAD  010008    77        LD   BC,#800        ;longueur bloc
4BB0  EDB0      78        LDIR                ;recopie
4BB2  C1        79        POP  BC
4BB3  10F2      80        DJNZ BCL4
                81 ;
                82 ;echange ecran <-> sauvegarde
                83 ;
4BB5  214461    84        LD   HL,24900       ;debut zone de transfert
4BB8  1100C0    85        LD   DE,#C000       ;debut ecran
4BBB  010040    86        LD   BC,#4000       ;longueur
                87 ;
4BBE  C5        88 BCL5:  PUSH BC
4BBF  1A        89        LD   A,(DE)
4BC0  46        90        LD   B,(HL)
4BC1  77        91        LD   (HL),A
4BC2  78        92        LD   A,B
4BC3  12        93        LD   (DE),A
4BC4  13        94        INC  DE
4BC5  23        95        INC  HL
4BC6  C1        96        POP  BC
4BC7  0B        97        DEC  BC
4BC8  78        98        LD   A,B
4BC9  B1        99        OR   C
```

```
4BCA  20F2      100         JR   NZ,BCL5
                101 ;
4BCC  C9        102         RET
                103 ;
                104 ;conversion coordonnees -> adresse ecran en mode 0
                105 ;
4BCD  CB3C      106 CONV:   SRL  H                  ;-> Y/2  (0<Y<200)
4BCF  CB1D      107         RR   L
4BD1  CB3A      108         SRL  D                  ;-> X/4  (0<X<160)
4BD3  CB1B      109         RR   E
4BD5  CB3A      110         SRL  D
4BD7  CB1B      111         RR   E
4BD9  CD1DBC    112         CALL #BC1D              ;->appel routine de conversion
4BDC  C9        113         RET
```

Les transformations dans une fenêtre

On définit à l'aide du joystick une fenêtre (position et taille), et on sélectionne une transformation en tapant une commande au clavier (ce dialogue est géré en assembleur) ; il y a alors branchement dans un des sous-modules de la routine FENETR.ASS :

- le module **INVV** réalise une inversion verticale à l'intérieur de la fenêtre de travail ;
- le module **INVH** réalise une inversion horizontale ;
- le module **INVC** inverse les couleurs par "inversion" d'un octet-écran.

Voici la liste de la **routine FENETR.ASS** (pour plus de détail sur l'utilisation de ce programme, lire les explications du guide utilisateur en annexe) ; ce programme fait appel au sous-programme de validité de déplacement VALID.ASS. Son utilisation en Basic nécessite la redéfinition du caractère 253 par l'instruction SYMBOL 253,255,129,129,129,129,129,129,255.

```
                1 ; ----  FENETRE : PROGRAMME PRINCIPAL  ----
                2 ;
4268            3         ORG  17000
4A9C            4 VALID:  EQU  19100              ;s/p deplacement
4AA2            5 LIMX:   EQU  #4AA2
4AA4            6 LIMY:   EQU  #4AA4
```

```
4268 1802       7         JR   DEBUT
426A FD         8 CAR:    DEFB 253              ;code car. curseur
426B 00         9 DRAP:   DEFB 0                ;drapeaux (bits 0 et 1)
426C 3E01      10 DEBUT:  LD   A,1
426E CD59BC    11         CALL #BC59            ;mode graph. XOR
4271 CD0A43    12         CALL AFFCAR           ;affichage caractere
               13 ;
4274 3E42      14 RELECT: LD   A,66             ;code touche ESC
4276 CD1EBB    15         CALL #BB1E            ;test touche
4279 C0        16         RET  NZ               ;retour si appuyee
427A CD24BB    17         CALL #BB24            ;lecture joystick
427D CB67      18         BIT  4,A
427F C4AA42    19         CALL NZ,CLIC          ;CLIC si feu
4282 3A6B42    20         LD   A,(DRAP)
4285 FE03      21         CP   3                ;pas test clavier
4287 2008      22         JR   NZ,DEPL          ;si coins non fixes
4289 CD1BBB    23         CALL #BB1B            ;touche appuyee?
428C DCE142    24         CALL C,CLAV           ;CLAV si oui
428F 18E3      25         JR   RELECT
4291 7C        26 DEPL:   LD   A,H              ;etat joystick 0
4292 E60F      27         AND  #0F              ;masque directions
4294 28DE      28         JR   Z,RELECT         ;relecture si rien
               29 ;
4296 CD9C4A    30         CALL VALID            ;deplacement
4299 38D9      31         JR   C,RELECT         ;relecture si deplacement
               32 ;                               impossible
429B E5        33         PUSH HL               ;sauv. nouvelle position
429C D5        34         PUSH DE
429D CD0443    35         CALL AFFCUR           ;effacement curseur
42A0 D1        36         POP  DE               ;rest. nouvelle position
42A1 E1        37         POP  HL
42A2 CDC0BB    38         CALL #BBC0            ;mise a jour pos. curseur
42A5 CD0443    39         CALL AFFCUR           ;affichage curseur
42A8 18CA      40         JR   RELECT
               41 ;
               42 ; --> Fixation des coins 0 et F de la fenetre
               43 ;
42AA CD24BB    44 CLIC:   CALL #BB24
42AD CB67      45         BIT  4,A              ;attend le relachement
42AF 20F9      46         JR   NZ,CLIC          ;de la touche "FEU"
42B1 CD0443    47         CALL AFFCUR           ;effacement curseur
42B4 216B42    48         LD   HL,DRAP
42B7 7E        49         LD   A,(HL)
42B8 1F        50         RRA                   ;si drapeau1=0
42B9 380F      51         JR   C,COINF          ;fixer coin 0
42BB CBC6      52 COINO:  SET  0,(HL)           ;drapeau1=1
42BD CDC6BB    53         CALL #BBC6            ;demande pos. curseur
42C0 ED535143  54         LD   (OX),DE          ;coordonnees dans
42C4 225343    55         LD   (OY),HL          ;OX et OY
```

```
42C7  C32743      56          JP   AFFFEN             ;affichage fenetre
42CA  1F          57 COINF:   RRA                     ;si drapeau2=0
42CB  380F        58          JR   C,FINFEN          ;fixer coin F
42CD  CBCE        59          SET  1,(HL)            ;drapeau2=1
42CF  CDC6BB      60          CALL #BBC6             ;demande pos. curseur
42D2  ED535543    61          LD   (FX),DE           ;coordonnees dans
42D6  225743      62          LD   (FY),HL           ;FX et FY
42D9  C32743      63          JP   AFFFEN            ;affichage fenetre
42DC  AF          64 FINFEN:  XOR  A
42DD  77          65          LD   (HL),A            ;RAZ drapeaux
42DE  C30A43      66          JP   AFFCAR            ;affichage caractere
                  67 ;
                  68 ; --> Test touches "H","V","C"
                  69 ;
42E1  F5          70 CLAV:    PUSH AF                ;sauve touche appuyee
42E2  CD2743      71          CALL AFFFEN            ;effacement fenetre
42E5  F1          72          POP  AF
42E6  E6DF        73          AND  %11011111         ;-> majuscule
42E8  FE48        74 TH:      CP   72                ;touche "H" ?
42EA  2005        75          JR   NZ,TV
42EC  CD0744      76          CALL INVH              ;inversion horizontale
42EF  1810        77          JR   FINCLA
42F1  FE56        78 TV:      CP   86                ;touche "V" ?
42F3  2005        79          JR   NZ,TC
42F5  CDC243      80          CALL INVV              ;inversion verticale
42F8  1807        81          JR   FINCLA
42FA  FE43        82 TC:      CP   67                ;touche "C" ?
42FC  2003        83          JR   NZ,FINCLA
42FE  CD4644      84          CALL INVC              ;inversion couleur
4301  C32743      85 FINCLA:  JP   AFFFEN            ;affichage fenetre
                  86
                  87 ; --> Affichage/effacement curseur
                  88 ;
4304  3A6B42      89 AFFCUR:  LD   A,(DRAP)
4307  A7          90          AND  A                 ;si 1 drapeau <> 0
4308  201D        91          JR   NZ,AFFFEN         ;affichage fenetre
430A  3A6A42      92 AFFCAR:  LD   A,(CAR)           ;n0 caractere
430D  CDFCBB      93          CALL #BBFC             ;affichage
4310  CDC6BB      94          CALL #BBC6             ;demande pos. curseur
4313  01E0FF      95          LD   BC,-32            ;restauration
4316  EB          96          EX   DE,HL             ;abscisse curseur
4317  09          97          ADD  HL,BC
4318  EB          98          EX   DE,HL
4319  CDC0BB      99          CALL #BBC0             ;positionne curseur
431C  3E0E       100          LD   A,14
431E  32A44A     101          LD   (LIMY),A          ;fixation des
4321  3E64       102          LD   A,#64             ;limites d'affichage
4323  32A24A     103          LD   (LIMX),A
4326  C9         104          RET
```

```
4327  CDC6BB     105 AFFFEN: CALL #BBC6              ;demande pos. curseur
432A  E5         106         PUSH HL
432B  D5         107         PUSH DE
432C  ED5B5143   108         LD   DE,(OX)           ;abscisse coin oppose
4330  D5         109         PUSH DE
4331  CDF6BB     110         CALL #BBF6             ;bord bas fenetre
4334  D1         111         POP  DE
4335  2A5343     112         LD   HL,(OY)           ;ordonnee coin oppose
4338  E5         113         PUSH HL
4339  CDF6BB     114         CALL #BBF6             ;bord gauche
433C  E1         115         POP  HL
433D  D1         116         POP  DE
433E  D5         117         PUSH DE
433F  CDF6BB     118         CALL #BBF6             ;bord haut
4342  D1         119         POP  DE
4343  E1         120         POP  HL
4344  CDF6BB     121         CALL #BBF6             ;bord droite
4347  AF         122         XOR  A
4348  32A44A     123         LD   (LIMY),A          ;fixation des
434B  3E80       124         LD   A,#80             ;limites d'affichage
434D  32A24A     125         LD   (LIMX),A
4350  C9         126         RET
                 127 ;
                 128 ; --------  INVERSIONS D'UNE FENETRE  --------
                 129 ;
                 130 ;inversions verticale,horizontale et couleur
                 131 ;d'une fenetre dont les coordonnees de deux
                 132 ;coins opposes se trouvent en OX,OY et FX,FY
                 133 ;
4351  4001       134 OX:     DEFW 320
4353  C800       135 OY:     DEFW 200
4355  7602       136 FX:     DEFW 630
4357  0400       137 FY:     DEFW 4
                 138 ;sous-programme DEB1 :
                 139 ;  entree: - OX,OY et FX,FY : 2 coins opposes
                 140 ;  sortie: - OX,OY : coin superieur gauche
                 141 ;          - FX,FY : coin inferieur droite
                 142 ;          - HL : adresse ecran de depart
                 143 ;          - B  : largeur fenetre
                 144 ;          - C  : hauteur fenetre
4359  ED4B5143   145 DEB1:   LD   BC,(OX)
435D  ED5B5543   146         LD   DE,(FX)
4361  62         147         LD   H,D
4362  6B         148         LD   L,E
4363  B7         149         OR   A
4364  ED42       150         SBC  HL,BC
4366  300A       151         JR   NC,SUIT1
4368  ED435543   152         LD   (FX),BC           ;FX=max(OX,FX)
436C  ED535143   153         LD   (OX),DE           ;OX=min(OX,FX)
```

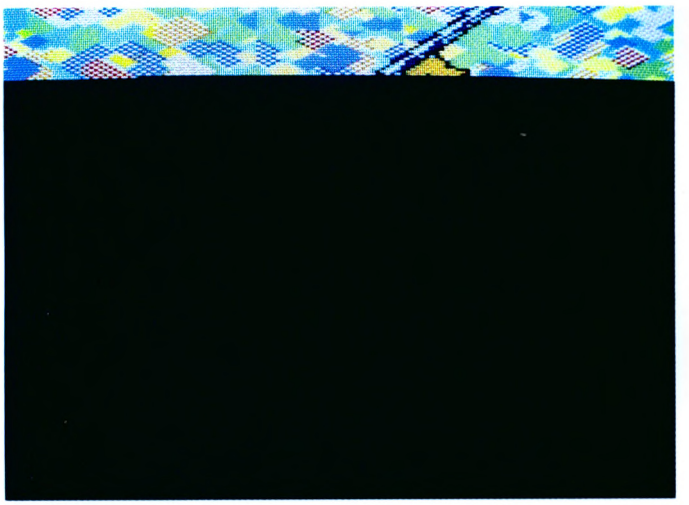
```
4370  18E7      154          JR   DEB1
4372  CB3C      155 SUIT1:   SRL  H                ;(FX-OX)/8
4374  CB1D      156          RR   L
4376  CB3C      157          SRL  H
4378  CB1D      158          RR   L
437A  CB3D      159          SRL  L
437C  7D        160          LD   A,L              ;largeur en octets
              161 ;
437D  ED4B5343  162 DEB2:    LD   BC,(OY)
4381  ED5B5743  163          LD   DE,(FY)
4385  60        164          LD   H,B
4386  69        165          LD   L,C
4387  B7        166          OR   A
4388  ED52      167          SBC  HL,DE
438A  300A      168          JR   NC,SUIT2
438C  ED435743  169          LD   (FY),BC          ;FY=min(OY,FY)
4390  ED535343  170          LD   (OY),DE          ;OY=max(OY,FY)
4394  18E7      171          JR   DEB2
4396  CB3C      172 SUIT2:   SRL  H                ;(OY-FY)/2
4398  CB1D      173          RR   L
439A  4D        174          LD   C,L              ;hauteur en lignes
439B  47        175          LD   B,A              ;largeur en octets
439C  04        176          INC  B
439D  0C        177          INC  C
439E  ED5B5143  178          LD   DE,(OX)
43A2  2A5343    179          LD   HL,(OY)
43A5  CDAC43    180          CALL CONV             ;HL=adresse ecran depart
43A8  C9        181          RET
              182 ;
              183 ;Conversion coordonnees - adresse ecran
              184 ;
43A9  CDC6BB    185 ADR:     CALL #BBC6
43AC  F5        186 CONV:    PUSH AF
43AD  C5        187          PUSH BC
43AE  CB3C      188          SRL  H                ;Y / 2
43B0  CB1D      189          RR   L
43B2  CB3A      190          SRL  D                ;X / 4
43B4  CB1B      191          RR   E
43B6  CB3A      192          SRL  D
43B8  CB1B      193          RR   E
43BA  CD1DBC    194          CALL #BC1D            ;conversion
43BD  C1        195          POP  BC
43BE  F1        196          POP  AF
43BF  C9        197          RET
              198 ;
              199 ; --------  INVERSION VERTICALE  --------
              200 ;
43C0  0100      201 LARG:    DEFW 1                ;stockage largeur
6144            202 SAUV:    EQU  24900            ;adresse sauvegarde ligne
```

```
                    203 ;
43C2  CD5943        204 INVV:   CALL DEB1
                    205 ;
43C5  E5            206         PUSH HL                 ;HL=adresse coin sup.gauche
43C6  ED5B5143      207         LD   DE,(OX)
43CA  2A5743        208         LD   HL,(FY)
43CD  CDAC43        209         CALL CONV
43D0  EB            210         EX   DE,HL              ;DE=adresse coin inf.gauche
43D1  E1            211         POP  HL
                    212 ;
43D2  CB39          213         SRL  C                  ;hauteur / 2
43D4  78            214         LD   A,B
43D5  32C043        215         LD   (LARG),A           ;stockage largeur
                    216 ;
43D8  C5            217 VER1:   PUSH BC
43D9  E5            218         PUSH HL
43DA  D5            219         PUSH DE
                    220 ;
43DB  114461        221         LD   DE,SAUV
43DE  ED4BC043      222         LD   BC,(LARG)
43E2  EDB0          223         LDIR                    ;ligne sup -> SAUV
                    224 ;
43E4  E1            225         POP  HL
43E5  D1            226         POP  DE
43E6  D5            227         PUSH DE
43E7  E5            228         PUSH HL
43E8  ED4BC043      229         LD   BC,(LARG)
43EC  EDB0          230         LDIR                    ;ligne inf -> ligne sup
                    231 ;
43EE  D1            232         POP  DE
43EF  D5            233         PUSH DE
43F0  214461        234         LD   HL,SAUV
43F3  ED4BC043      235         LD   BC,(LARG)
43F7  EDB0          236         LDIR                    ;SAUV -> ligne inf
                    237 ;
                    238 ;
43F9  E1            239         POP  HL
43FA  CD29BC        240         CALL #BC29              ;ligne inf precedente
43FD  EB            241         EX   DE,HL
43FE  E1            242         POP  HL
43FF  CD26BC        243         CALL #BC26              ;ligne sup suivante
4402  C1            244         POP  BC
4403  0D            245         DEC  C
4404  20D2          246         JR   NZ,VER1            ;test fin fenetre
4406  C9            247         RET
                    248 ;
                    249 ; --------  INVERSION HORIZONTALE  --------
                    250 ;
4407  CD5943        251 INVH:   CALL DEB1
```

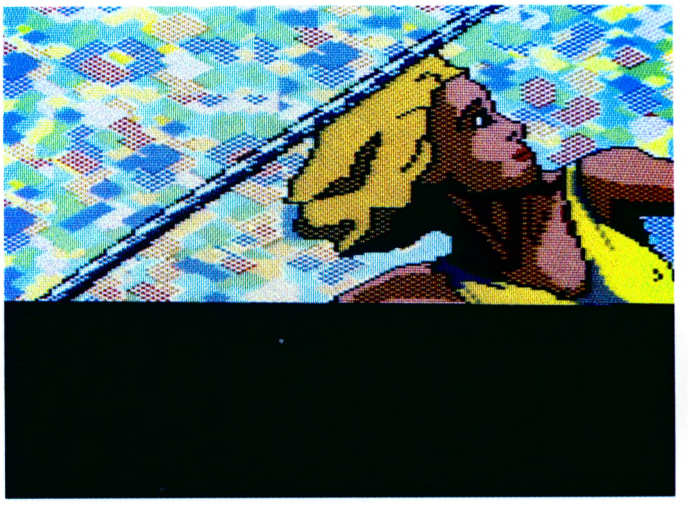
```
440A  CB38      252         SRL  B                ;largeur / 2
440C  E5        253         PUSH HL               ;HL=adr coin sup gauche
440D  ED5B5543  254         LD   DE,(FX)
4411  2A5343    255         LD   HL,(OY)
4414  CDAC43    256         CALL CONV
4417  EB        257         EX   DE,HL            ;DE=adr coin sup droite
4418  E1        258         POP  HL
                259 ;
4419  C5        260 HOR1:   PUSH BC
441A  E5        261         PUSH HL
441B  D5        262         PUSH DE
                263 ;
441C  1A        264 HOR2:   LD   A,(DE)
441D  CD3944    265         CALL RTR              ;inversion (DE)
4420  4F        266         LD   C,A
4421  7E        267         LD   A,(HL)
4422  CD3944    268         CALL RTR              ;inversion (HL)
4425  12        269         LD   (DE),A
4426  71        270         LD   (HL),C           ;echange (HL)-(DE)
4427  23        271         INC  HL
4428  1B        272         DEC  DE
4429  10F1      273         DJNZ HOR2             ;test fin ligne
                274 ;
442B  E1        275         POP  HL
442C  CD26BC    276         CALL #BC26            ;ligne ecran suivante
442F  EB        277         EX   DE,HL
4430  E1        278         POP  HL
4431  CD26BC    279         CALL #BC26            ;ligne ecran suivante
4434  C1        280         POP  BC
4435  0D        281         DEC  C
4436  20E1      282         JR   NZ,HOR1          ;test fin fenetre
4438  C9        283         RET
                284 ;
                285 ;Inversion octet ecran contenu dans A
                286 ;   <=> echange point droite - point gauche
                287 ;
4439  C5        288 RTR:    PUSH BC
443A  47        289         LD   B,A
443B  E6AA      290         AND  #AA              ;point gauche
443D  1F        291         RRA                   ;devient point droite
443E  4F        292         LD   C,A
443F  78        293         LD   A,B
4440  E655      294         AND  #55              ;point droite
4442  17        295         RLA                   ;devient point gauche
4443  B1        296         OR   C                ;combinaison des 2
4444  C1        297         POP  BC
4445  C9        298         RET
                299 ;
```

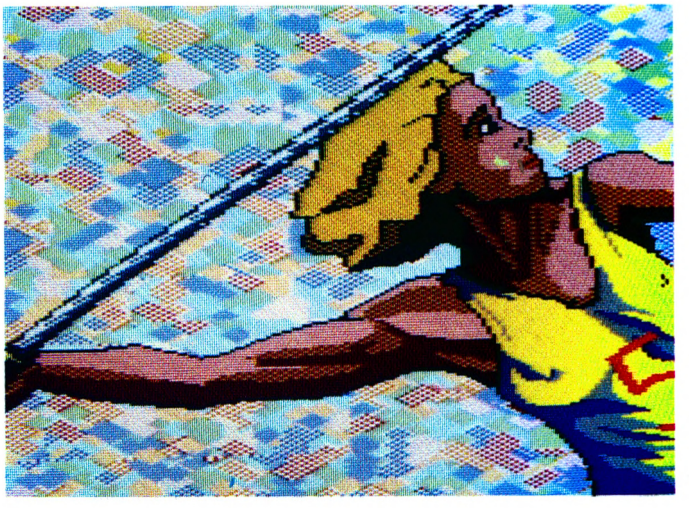
```
                    300 ; -------- INVERSION DES COULEURS --------
                    301 ;
4446  CD5943        302 INVC:   CALL DEB1
                    303 ;
4449  C5            304 COUL1:  PUSH BC
444A  E5            305         PUSH HL
                    306 ;
444B  7E            307 COUL2:  LD   A,(HL)
444C  2F            308         CPL                     ;inversion octet ecran
444D  77            309         LD   (HL),A
444E  23            310         INC  HL
444F  10FA          311         DJNZ COUL2              ;test fin ligne
                    312 ;
4451  E1            313         POP  HL
4452  CD26BC        314         CALL #BC26              ;ligne ecran suivante
4455  C1            315         POP  BC
4456  0D            316         DEC  C
4457  20F0          317         JR   NZ,COUL1           ;test fin fenetre
4459  C9            318         RET
```

ANIMATIONS

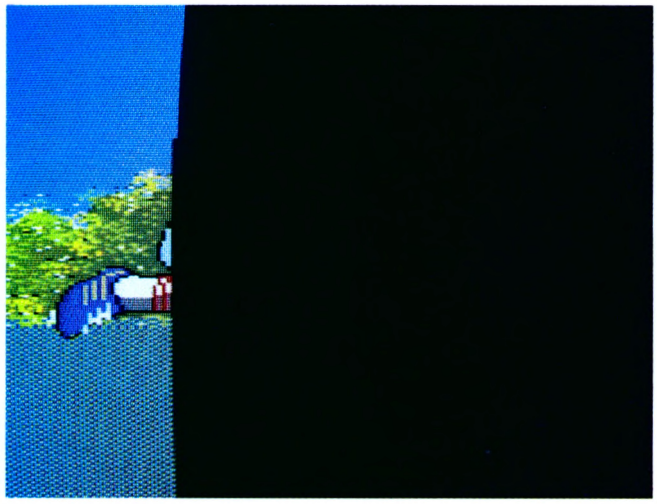


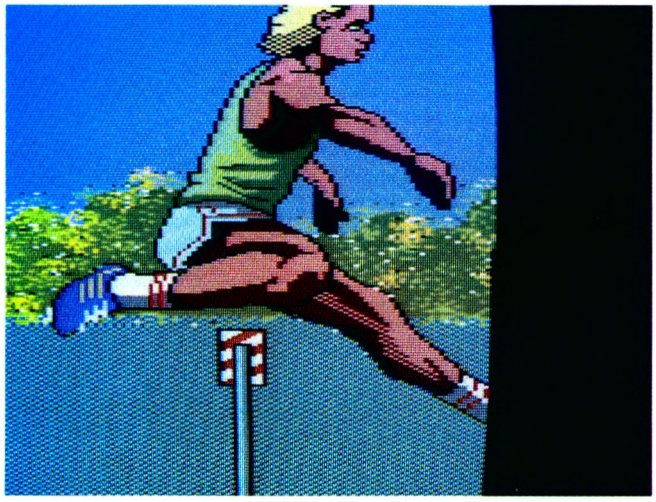
Affichage ligne à ligne

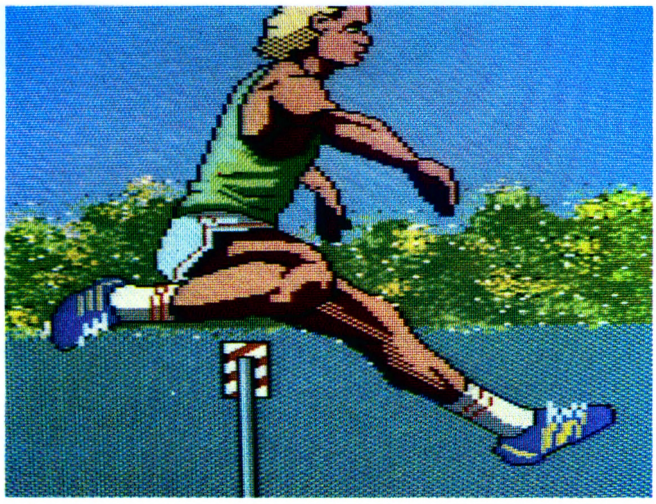


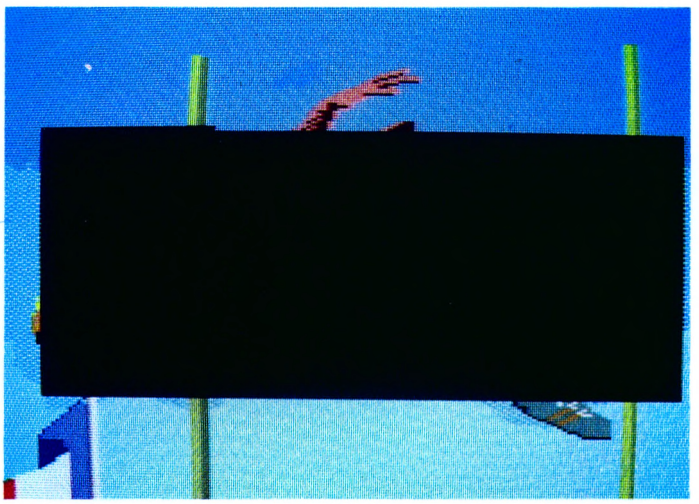


Affichage par groupe de colonnes

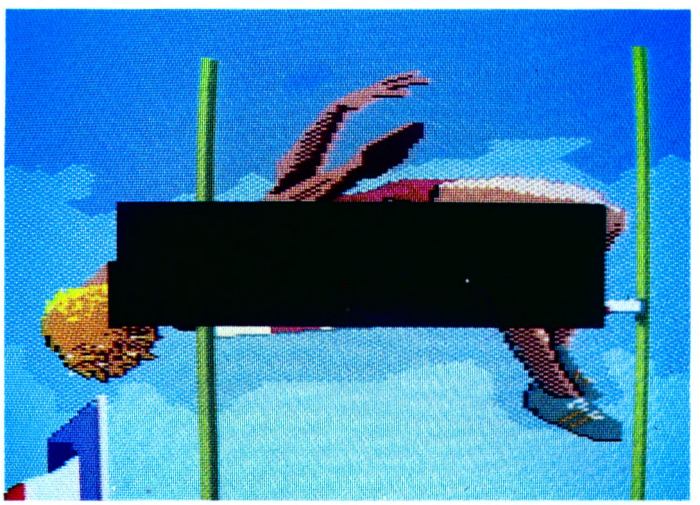


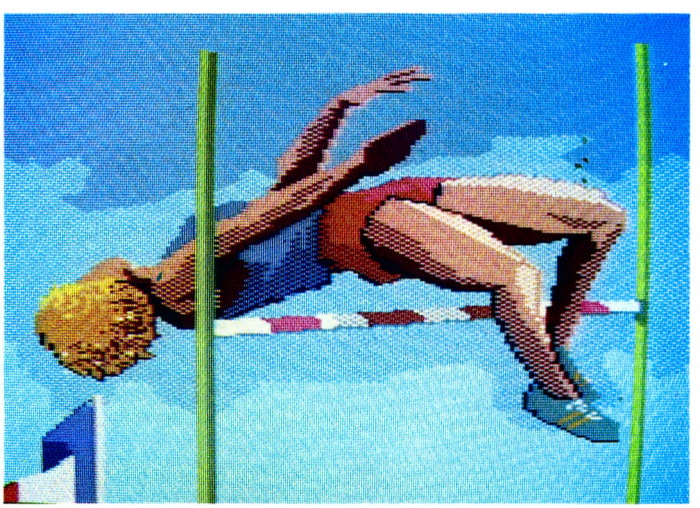


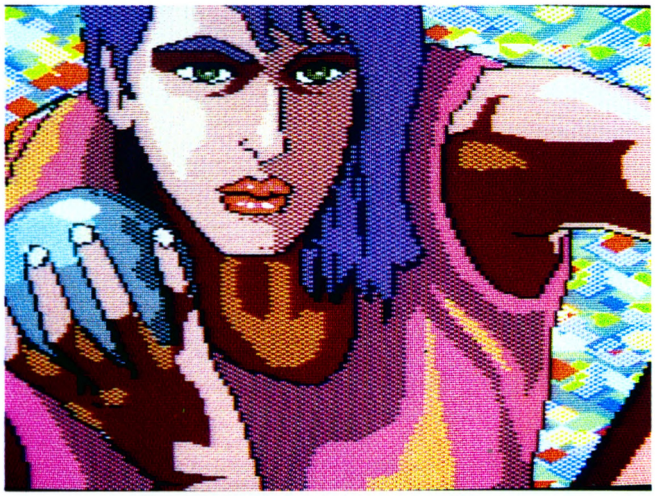




Affichage d'un serpentin

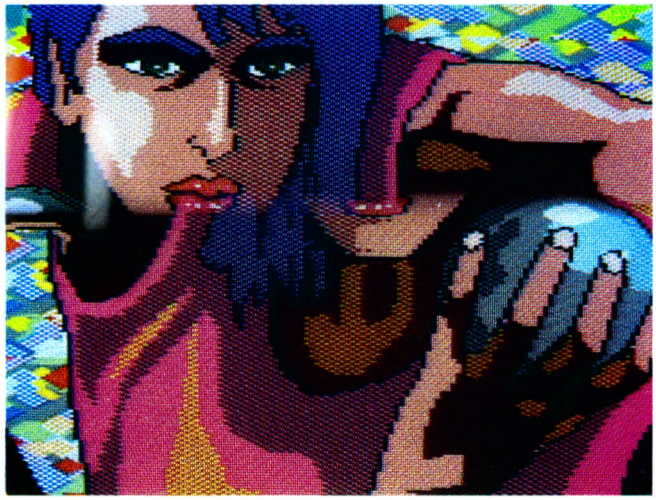


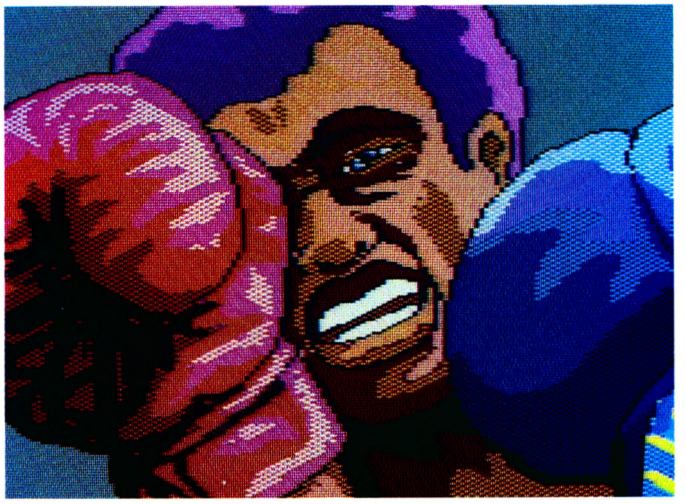


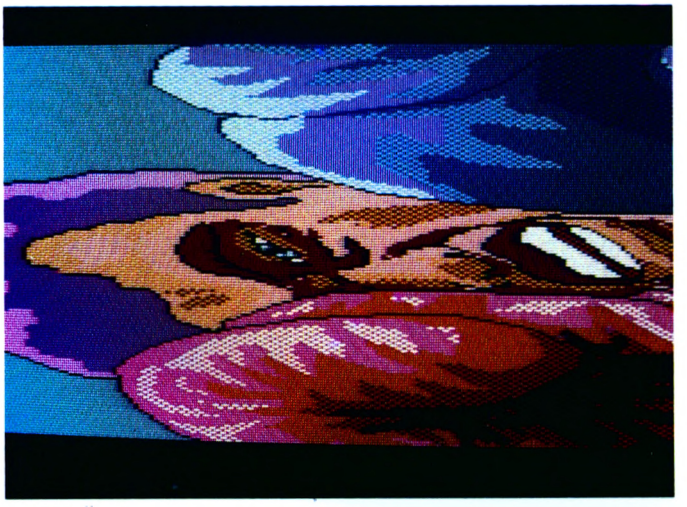


Inversion de couleur

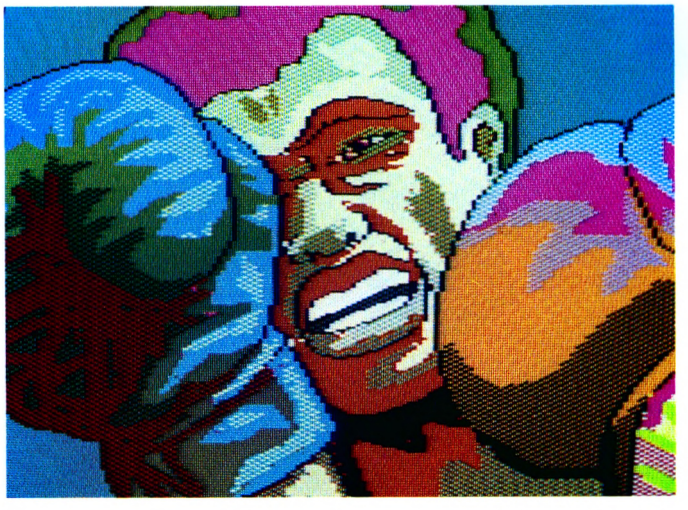
Inversion horizontale de l'image en cours







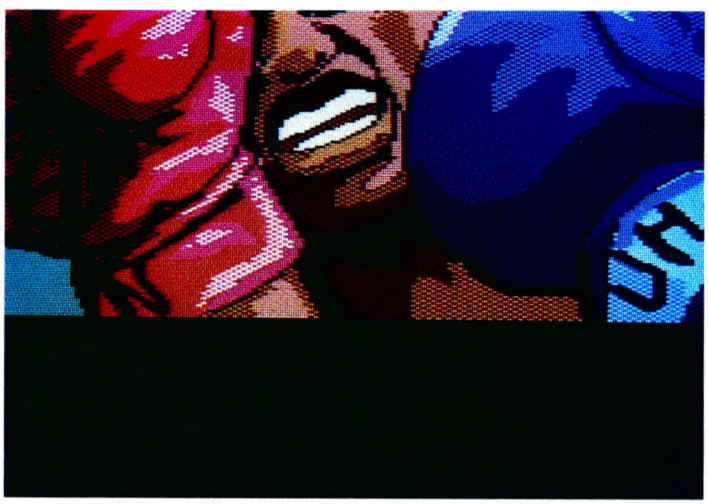
Renversement d'une partie de l'image

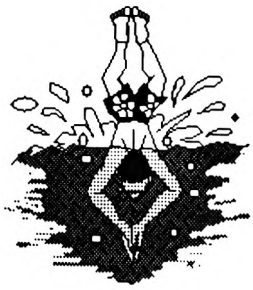


Changement des couleurs

Animation
de cinq pages graphiques

Scrolling descendant



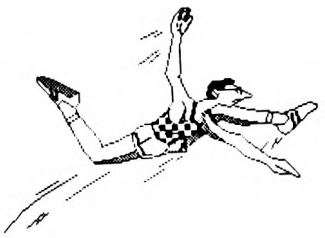


Introduction

Les animations proposées exploitent le déplacement – et le découpage – d'images, ou de portions d'images dans la mémoire ; l'utilisation conjointe de plusieurs pages graphiques, si vous disposez de 128K de mémoire, permet d'obtenir d'excellents résultats ; la richesse des effets visuels étant liée à la qualité des images créées.

Ces animations sont applicables à vos propres images ; nous vous décrivons quelques exemples ; il y a bien d'autres possibilités de faire évoluer les images, à vous de les imaginer ; certains exemples peuvent être détournés de leur fonction initiale : ainsi le découpage d'une image en rectangles peut déboucher sur la réalisation d'un puzzle ; il vous suffit d'afficher les morceaux constituants de l'image dans le désordre (processus aléatoire), à vous de reconstituer l'image de départ à l'aide de la souris par exemple...

Ce chapitre constitue une boîte à outils idéale pour la fabrication d'une "bande dessinée électronique", mais attention, si vous pouvez donner libre cours à votre imagination, il ne faudra pas être trop exigeant, et exploiter au mieux les possibilités techniques offertes par votre micro-ordinateur.



Chapitre 1

LE CHAMP D'APPLICATION ET SES LIMITES

Il n'est pas envisageable de faire évoluer des images couleurs en trois dimensions ; cependant, une série d'images exploitant avec habileté les lois de la perspective peut donner l'impression d'une animation 3D ; il s'agit cependant d'un travail artisanal puisque les "angles de vue" générés sont créés par le dessinateur et non calculés par le processeur.

La construction d'une animation est conditionnée par les possibilités de stockage et par les vitesses de traitement.

LES POSSIBILITÉS DE STOCKAGE

Le nombre d'images "pleine page" que l'on peut stocker en mémoire vive est réduit, même avec 128K (l'utilisation du deuxième banc auxiliaire permet de stocker cinq pages graphiques) ; on peut envisager de stocker des fenêtres d'image si une animation n'affecte qu'une zone de l'image par exemple ; l'utilisation de fenêtres permet d'éviter des redondances de stockage.

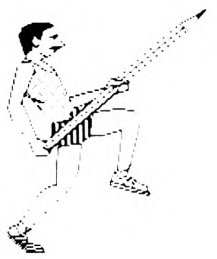
D'autre part, la capacité du support externe de stockage (une disquette en général) restreint le nombre total d'images utilisé dans l'animation ; il est utile de construire une routine de compactage/

décompactage d'un fichier image afin d'optimiser la place sur la disquette. Ce procédé est souvent employé dans les jeux d'aventures afin d'éviter au joueur des manipulations de disquettes.

LES VITESSES DE TRAITEMENT

Les vitesses de traitement sont conditionnées par la nature de l'animation, la solution choisie pour la programmer ; il est parfois nécessaire de temporiser l'animation.

La vitesse de chargement d'un programme et d'une image est dépendante du mode d'organisation des données sur le support magnétique et des caractéristiques des routines d'entrée-sortie du système d'exploitation.



Chapitre 2

LES PROGRAMMES UTILISANT DES TABLES D'ADRESSES

LE PRINCIPE DE FONCTIONNEMENT LA DESCRIPTION DES TABLES

Le principe des animations proposées consiste à construire des tables d'adresses indépendantes des modules d'animation proprement dit ; le module d'animation exploite (lecture) le contenu de ces tables.

L'intérêt de cette construction modulaire répond aux objectifs suivants :

- pouvoir développer rapidement différents modes d'animation, sans intervenir sur le module de traitement principal ; la recherche d'effets d'animation consistant à effectuer des variations sur le contenu des tables ;
- optimiser les vitesses de traitement ; en effet, le module d'animation est déchargé des calculs d'adresse et n'effectue que des lectures des valeurs-adresses.

Chacune des réalisations proposées est ainsi conçue :

- un programme de traitement d'affichage (écrit en assembleur) exploite une ou deux tables d'adresses ;

– plusieurs modules (écrits en Basic et en assembleur) servent à construire les tables.
Les modules Basic de construction de tables sont exécutés une fois pour toutes, et la table créée en mémoire vive est sauvegardée ; cette méthode est facile à mettre en œuvre, mais le chargement d'une table peut occasionner une attente néfaste entre deux animations distinctes.
Les modules Assembleur sont exécutés avant chaque animation, sauf si la table existe déjà en mémoire ; la place nécessaire en mémoire est plus importante dans ce cas (table + programme), mais l'on gagne en souplesse d'utilisation et en vitesse d'exécution ; de plus, on économise de la place sur la disquette de travail.

Les tables d'adresses construites ont les caractéristiques suivantes :

Nom	Adresse de début	Longueur	Description
T1	20100	400	Adresses des lignes de l'écran de travail masqué
T2	20500	400	Adresses des lignes de l'écran visible
T3	20900	4000	Adresses des 2000 rectangles (correspondant à l'écran visible)

Rappelons que l'adresse de début de l'écran visible est 49152 (ou C000 en hexadécimal) ; l'adresse de début de l'écran de travail masqué est fixée à 24900 (juste après la table T3).

L'utilitaire Assembleur INVTAB.ASS, que nous présentons ci-après, permet d'inverser le contenu d'une table, le programme à trois points d'entrée correspondant à une des tables T1, T2 ou T3. Après exécution de ce module, on pourra sauvegarder la table construite en vue d'une utilisation ultérieure.

Voici la liste du **module Assembleur INVTAB.ASS :**

```
                    1 ; --------  INVERSION DES TABLES T1,T2 ET T3  --------
                    2 ;
                    3 ;                (par mots de 16 bits)
                    4 ;
4C2C                5          ORG  19500
4C2C  21844E        6 INVT1:   LD   HL,20100          ;debut T1
4C2F  111250        7          LD   DE,20498          ;fin T1
```

```
4C32  016400      8         LD   BC,100          ;nbre adr.de T1 /2
4C35  1814        9         JR   BCL
                  10 ;
4C37  211450      11 INVT2: LD   HL,20500        ;debut T2
4C3A  11A251      12        LD   DE,20898        ;fin T2
4C3D  016400      13        LD   BC,100          ;nbre adr.de T2 /2
4C40  1809        14        JR   BCL
                  15 ;
4C42  21A451      16 INVT3: LD   HL,20900        ;debut T3
4C45  114261      17        LD   DE,24898        ;fin T3
4C48  01E803      18        LD   BC,1000         ;nbre adr.de T3 /2
                  19 ;
4C4B  C5          20 BCL:   PUSH BC
                  21 ;
4C4C  46          22        LD   B,(HL)
4C4D  1A          23        LD   A,(DE)
4C4E  77          24        LD   (HL),A          ;echange octet pds faible
4C4F  78          25        LD   A,B
4C50  12          26        LD   (DE),A
                  27 ;
4C51  23          28        INC  HL
4C52  13          29        INC  DE
                  30 ;
4C53  46          31        LD   B,(HL)
4C54  1A          32        LD   A,(DE)
4C55  77          33        LD   (HL),A          ;echange octet pds fort
4C56  78          34        LD   A,B
4C57  12          35        LD   (DE),A
                  36 ;
4C58  23          37        INC  HL              ;adresses suivantes
4C59  1B          38        DEC  DE
4C5A  1B          39        DEC  DE
4C5B  1B          40        DEC  DE
                  41 ;
4C5C  C1          42        POP  BC
4C5D  0B          43        DEC  BC
4C5E  78          44        LD   A,B
4C5F  B1          45        OR   C
4C60  20E9        46        JR   NZ,BCL          ;test fin table
4C62  C9          47        RET
```

LES "SCROLLINGS"

Un scroll est un rouleau servant de support à un document écrit, un papyrus par exemple ; la lecture de notre premier papyrus électronique consiste en une translation de l'image du haut vers le bas de l'écran.

Le **programme Basic TSCROLL** construit la table "tscroll", qui est la concaténation des deux tables T1 et T2 ; voici la liste de ce programme :

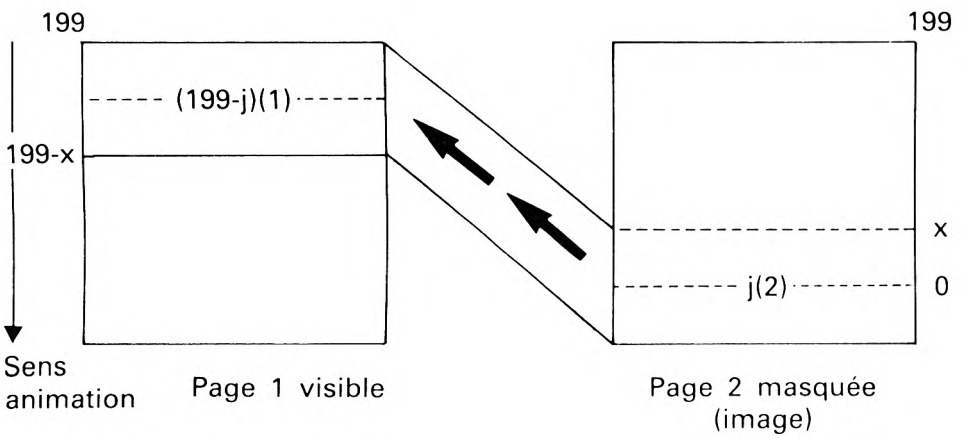
```
100 '---- Initialisation table adresses lignes pour scrolling descendant ----
110 '
120 ' Debut table adresses lignes ecran cache : 20100
130 ' Debut table adresses lignes ecran       : 20500
140 ' Longueur de chaque table                :   400
150 '
160 CLS:MEMORY 20000:cpt=20500
170 FOR a=0 TO 24
180     FOR i=0 TO 7
190         adr1=49152+a*80+i*2048     '-> adresse ligne ecran
200         adr2=24900+a*80+i*2048     '-> adresse ligne ecran cache
210         POKE cpt,adr1 MOD 256:POKE cpt+1,INT(adr1/256)
220         POKE cpt-400,adr2 MOD 256:POKE cpt-399,INT(adr2/256)
230         cpt=cpt+2
240     NEXT i
250 NEXT a
260 SAVE "tscroll",b,20100,800
```

L'autre méthode consiste à exécuter le **module Assembleur INT1T2.ASS** qui initialise également les tables T1 et T2 en vue du scrolling descendant ; voici la liste de ce module Assembleur :

```
                    1 ; -------- INITIALISATION TABLES T1 ET T2 --------
                    2 ;
4DBC                3         ORG  19900
4DBC  DD21844E      4         LD   IX,20100          ;debut T1
4DC0  FD211450      5         LD   IY,20500          ;debut T2
4DC4  2100C0        6         LD   HL,#C000          ;debut ecran
4DC7  114461        7         LD   DE,24900          ;debut ecran cache
4DCA  01C800        8         LD   BC,200            ;nombre de lignes
                    9 ;
4DCD  DD7300       10 BCL:    LD   (IX+0),E          ;chargement T1
4DD0  DD7201       11         LD   (IX+1),D
4DD3  FD7500       12         LD   (IY+0),L          ;chargement T2
4DD6  FD7401       13         LD   (IY+1),H
4DD9  DD23         14         INC  IX
4DDB  DD23         15         INC  IX
4DDD  FD23         16         INC  IY
4DDF  FD23         17         INC  IY
```

```
                    18 ;
4DE1  CD26BC        19          CALL #BC26              ;ligne ecran suivante
4DE4  EB            20          EX   DE,HL
4DE5  2144A1        21          LD   HL,-24252          ;decalage ecran-ecran cache
4DE8  19            22          ADD  HL,DE              ;ligne ecran cache suivante
4DE9  EB            23          EX   DE,HL
                    24 ;
4DEA  0B            25          DEC  BC                 ;test fin ecran
4DEB  78            26          LD   A,B
4DEC  B1            27          OR   C
4DED  20DE          28          JR   NZ,BCL
4DEF  C9            29          RET
```

Le module Assembleur de traitement d'affichage SCROLL.ASS réalise l'animation en lisant le contenu des tables ; le traitement d'une étape de l'animation peut être ainsi schématisé :



Etat d'avancement "x" du scrolling.

La page "1" est la page visible dans laquelle se déroule l'animation ; la page "2" est la page masquée contenant l'image à animer ; le *schéma ci-dessus* représente l'état d'avancement "x" de l'animation ; afin de décrire l'algorithme du traitement, nous employons la notation suivante :

– j(n) signifie ligne j de la page n.

Le traitement de l'étape "x" est le suivant :

- de j = 0 à x, écrire le contenu de j(2) ; dans (199-J)(1)
- faire j suivant.

Le traitement complet de la page est donc :

- de x = 199 à 0, incrémenter de (−1), effectuer le traitement de l'étape "x" ;
- faire x suivant.

Lorsque x diminue, la boucle interne de traitement est plus longue ; la vitesse du scrolling est variable, l'affichage étant ralenti au fur et à mesure de la progression de l'image.

Voici la liste du **programme Assembleur SCROLL.ASS** :

```
                    1 ;----   SCROLLING   ----
                    2 ;
                    3 ;TABLE T1:
                    4 ;   table des adresses des lignes
                    5 ;   de l'ecran cache
                    6 ;TABLE T2:
                    7 ;   table des adresses des lignes
                    8 ;   de l'ecran
                    9 ;
                   10 ;           debut    fin    long
                   11 ;
                   12 ;TABLE T1    20100  20499     400
                   13 ;TABLE T2    20500  20899     400
                   14 ;
4B64               15         ORG  19300
4B64  3E01         16         LD   A,1
4B66  111550       17         LD   DE,20501          ;1ere adresse de T2
                   18 ;
4B69  F5           19 SCROL1: PUSH AF
4B6A  D5           20         PUSH DE
4B6B  211350       21         LD   HL,20499          ;derniere adresse de T1
                   22 ;
4B6E  46           23 SCROL2: LD   B,(HL)
4B6F  2B           24         DEC  HL
4B70  4E           25         LD   C,(HL)            ;->BC=adresse ligne ecran cache courante
4B71  2B           26         DEC  HL
4B72  E5           27         PUSH HL                ;->ligne precedente
4B73  EB           28         EX   DE,HL
4B74  56           29         LD   D,(HL)
4B75  2B           30         DEC  HL
4B76  5E           31         LD   E,(HL)            ;->DE=adresse ligne ecran courante
4B77  2B           32         DEC  HL
4B78  E5           33         PUSH HL                ;->ligne precedente
4B79  60           34         LD   H,B
4B7A  69           35         LD   L,C
4B7B  015000       36         LD   BC,80
```

```
4B7E  EDB0      37        LDIR                ;->transfert
4B80  D1        38        POP  DE
4B81  E1        39        POP  HL
4B82  3D        40        DEC  A
4B83  20E9      41        JR   NZ,SCROL2
                42 ;
4B85  D1        43        POP  DE
4B86  13        44        INC  DE
4B87  13        45        INC  DE             ;descente d'un cran ds T2
4B88  F1        46        POP  AF
4B89  3C        47        INC  A
4B8A  FEC9      48        CP   201
4B8C  20DB      49        JR   NZ,SCROL1
4B8E  C9        50        RET
```

Si l'image à animer comporte du texte, un scrolling ascendant sera plus approprié pour la lecture ; dans ce cas, il faut inverser le contenu de nos deux tables T1 et T2 (utiliser pour cela l'utilitaire INVTAB.ASS) ; l'inversion du contenu de la table T1 seule donne un scrolling descendant de l'image inversée ; en inversant la table T2 seule, on obtient un scrolling ascendant de l'image inversée.

Vous êtes convaincus de l'intérêt d'écrire des applications modulaires, ce qui nous permet maintenant d'inventer d'autres types de scrollings et balayages ; pourquoi ne pas utiliser une table (T1) des adresses générées aléatoirement ? Vous imaginez l'effet produit ? A vous de jouer...

LA DÉCOMPOSITION D'UNE IMAGE EN 2 000 RECTANCLES !

Un rectangle élémentaire correspond en mode "0" à 8 lignes (hauteur) de 2 points (largeur), soit 8 octets de code ; ce qui est équivalent à un caractère en mode "2".

Chaque rectangle est identifié par l'adresse "mémoire-écran" de son octet supérieur ; ces adresses sont codées sur 2 octets et stockées dans la table T3 (cette table a donc une taille de 2 × 2 000 = 4 000 octets).

Le programme d'affichage RECTAN.ASS lit cette table séquentiellement ; après chaque lecture d'une adresse, on calcule l'adresse

correspondante dans l'écran masqué (soit "adresse"-2252 ; 2252 étant la valeur de décalage entre l'écran masqué et l'écran visible), et on affiche le rectangle correspondant.

Voici la liste du **module principal RECTAN.ASS** (ce module possède deux points d'entrée : &4BC8 pour l'affichage des rectangles et &4BF9 pour la remise à zéro de l'écran masqué) :

```
                        1 ; --------   RECTANGLES   --------
                        2 ;
                        3 ;Echange Ecran <-> Zone sauvegarde par petits rectangles (1x8)
                        4 ;
                        5 ;Zones memoire utilisees:
                        6 ;    NOM      DEBUT     LONGUEUR      DESCRIPTION
                        7 ; Tabrec      20900         4000      Adresses ecran rectangl
                        8 ; Sauv        24900        16384      Sauvegarde ecran
                        9 ;
                       10 ;Sous-programmes:
                       11 ;    NOM      DESCRIPTION
                       12 ; Raz         Initialisation de Sauv (a 0 ou autre)
                       13 ;
4BC8                   14         ORG  19400
4BC8  21A451           15 ECH:    LD   HL,20900           ;debut TABREC
4BCB  01D007           16         LD   BC,2000            ;nombre de rectangles
                       17 ;
4BCE  C5               18 BCL1:   PUSH BC
4BCF  5E               19         LD   E,(HL)
4BD0  23               20         INC  HL
4BD1  56               21         LD   D,(HL)             ;DE=adresse ecran du rectangle
4BD2  23               22         INC  HL
4BD3  E5               23         PUSH HL
4BD4  0608             24         LD   B,8                ;hauteur du rectangle
4BD6  2144A1           25         LD   HL,#A144           ;decalage Ecran-Sauv (-24252)
4BD9  19               26         ADD  HL,DE              ;HL=adresse sauvegarde rectangle
                       27 ;
4BDA  1A               28 BCL2:   LD   A,(DE)
4BDB  4E               29         LD   C,(HL)
4BDC  77               30         LD   (HL),A             ; } echange ecran <-> sauvegarde
4BDD  79               31         LD   A,C
4BDE  12               32         LD   (DE),A
                       33 ;
4BDF  3E08             34         LD   A,8
4BE1  84               35         ADD  A,H
4BE2  67               36         LD   H,A
4BE3  3E08             37         LD   A,8                ; } lignes suivantes
4BE5  82               38         ADD  A,D
4BE6  57               39         LD   D,A
                       40 ;
```

```
4BE7  10F1      41        DJNZ BCL2            ;test fin rectangle
                42 ;
4BE9  216400    43        LD   HL,100
4BEC  2B        44 BA:    DEC  HL              ;temporisation
4BED  7C        45        LD   A,H
4BEE  B5        46        OR   L
4BEF  20FB      47        JR   NZ,BA
                48 ;
4BF1  E1        49        POP  HL              ;restauration pointeur TABREC
4BF2  C1        50        POP  BC              ;restauration compteur
4BF3  0B        51        DEC  BC
4BF4  78        52        LD   A,B
4BF5  B1        53        OR   C
4BF6  20D6      54        JR   NZ,BCL1         ;test fin table
4BF8  C9        55        RET
                56 ;
4BF9  214461    57 RAZ:   LD   HL,24900        ;adresse Sauv
4BFC  010040    58        LD   BC,#4000        ;longueur
4BFF  3600      59 LOOP:  LD   (HL),0          ;initialisation a 0
4C01  23        60        INC  HL              ;octet suivant
4C02  0B        61        DEC  BC
4C03  79        62        LD   A,C
4C04  B0        63        OR   B
4C05  20F8      64        JR   NZ,LOOP         ;test fin Sauv
4C07  C9        65        RET
```

Les serpentins

Les animations "serpentin" correspondent à un affichage ordonné et continu des rectangles ; nous proposons trois types de parcours du serpentin, et donc six programmes de construction de la table T3 (trois programmes BASIC et trois modules assembleur) ; pour chacun de ces types, l'inversion de la table T3 (programme INVTAB.ASS) permet de changer le sens de parcours du serpentin (on a donc au total six animations distinctes).

Le tableau suivant résume les familles de parcours proposées :

Type de parcours du serpentin	Pgm BASIC pour construction de T3	Assembleur
De l'extérieur vers le centre	RECSER	RECTIN
De la gauche vers la droite	RECEDR	RECGD
Du haut vers le bas	RECHOBA	RECHB

Voici les listes de ces **six programmes :**

```
                    1 ; ---- INITIALISATION T3: SERPENTIN HAUT->BAS ----
                    2 ;
4C90                3         ORG  19600
4C90  21A451        4         LD   HL,20900          ;debut T3
4C93  1100C0        5         LD   DE,#C000          ;debut ecran
4C96  0E0C          6         LD   C,12
                    7 ;
4C98  0650          8 BCL:    LD   B,80              ;longueur d'une ligne
                    9 ;
4C9A  73           10 GD:     LD   (HL),E            ;chargement T3
4C9B  23           11         INC  HL
4C9C  72           12         LD   (HL),D
4C9D  23           13         INC  HL
                   14 ;
4C9E  13           15         INC  DE                ;parcours gauche-droite
4C9F  10F9         16         DJNZ GD
                   17 ;
4CA1  7B           18         LD   A,E
4CA2  C64F         19         ADD  A,79              ;ligne suivante
4CA4  5F           20         LD   E,A
4CA5  3001         21         JR   NC,S1
4CA7  14           22         INC  D
                   23 ;
4CA8  0650         24 S1:     LD   B,80              ;longueur d'une ligne
                   25 ;
4CAA  73           26 DG:     LD   (HL),E            ;chargement T3
4CAB  23           27         INC  HL
4CAC  72           28         LD   (HL),D
4CAD  23           29         INC  HL
                   30 ;
4CAE  1B           31         DEC  DE                ;;parcours droite-gauche
4CAF  10F9         32         DJNZ DG
                   33 ;
4CB1  7B           34         LD   A,E               ;ligne suivante
4CB2  C651         35         ADD  A,81
4CB4  5F           36         LD   E,A
4CB5  3001         37         JR   NC,S2
4CB7  14           38         INC  D
                   39 ;
4CB8  0D           40 S2:     DEC  C
4CB9  20DD         41         JR   NZ,BCL            ;test fin boucle
                   42 ;
4CBB  0650         43         LD   B,80              ;longueur derniere ligne
4CBD  73           44 DL:     LD   (HL),E            ;chargement table
4CBE  23           45         INC  HL
```

```
4CBF  72        46        LD   (HL),D
4CC0  23        47        INC  HL
4CC1  13        48        INC  DE                ;parcours gauche-droite
4CC2  10F9      49        DJNZ DL                ;test fin derniere ligne
4CC4  C9        50        RET
```

```
100 'RECHOBA.BAS
110 '
120 ' ---- Initialisation table adresses rectangles ----
130 '
140 '            SERPENTIN   (haut->bas)
150 '
160 MEMORY 20899
170 j=1:MODE 2:cpt=20900
180 '
190 FOR k=1 TO 23 STEP 2
200     FOR i=1 TO 80:GOSUB 310:NEXT i          'droite -> gauche
210     FOR i=80 TO 1 STEP -1:GOSUB 320:NEXT i  'gauche -> droite
220 NEXT k
230 '
240 FOR i=1 TO 80:GOSUB 310:NEXT i
250 '
260 SAVE "rectab",b,20900,4000
270 END
280 '
290 'calcul adresse ecran du rectangle et mise a jour table
300 '
310 adr=49152+(k-1)*80+i-1:GOTO 330
320 adr=49152+k*80+i-1
330 POKE cpt,adr-INT(adr/256)*256
340 POKE cpt+1,INT(adr/256)
350 cpt=cpt+2
360 RETURN
```

```
                1 ; ---- INITIALISATION T3: SERPENTIN GAUCHE-DROITE ----
                2 ;
4CF4            3         ORG  19700
4CF4  1100C0    4         LD   DE,#C000          ;debut ecran
4CF7  21A451    5         LD   HL,20900          ;debut T3
4CFA  0628      6         LD   B,40              ;nbre colonnes/2
                7 ;
4CFC  C5        8 BCL:    PUSH BC
                9 ;
4CFD  0619      10        LD   B,25              ;nbre lignes
```

```
                    11 ;
4CFF  73            12 HB:    LD   (HL),E          ;chargement T3
4D00  23            13        INC  HL
4D01  72            14        LD   (HL),D
4D02  23            15        INC  HL
                    16 ;
4D03  3E50          17        LD   A,80            ;ligne suivante:
4D05  83            18        ADD  A,E             ;->parcours haut-bas
4D06  5F            19        LD   E,A
4D07  3001          20        JR   NC,S1
4D09  14            21        INC  D
                    22 ;
4D0A  10F3          23 S1:    DJNZ HB
                    24 ;
4D0C  EB            25        EX   DE,HL
4D0D  01B1FF        26        LD   BC,-79
4D10  09            27        ADD  HL,BC           ;colonne suivante
4D11  EB            28        EX   DE,HL
                    29 ;
4D12  0619          30        LD   B,25            ;nombre de lignes
                    31 ;
4D14  73            32 BH:    LD   (HL),E          ;chargement T3
4D15  23            33        INC  HL
4D16  72            34        LD   (HL),D
4D17  23            35        INC  HL
                    36 ;
4D18  7B            37        LD   A,E             ;ligne precedente
4D19  D650          38        SUB  80              ;->parcours bas-haut
4D1B  5F            39        LD   E,A
4D1C  3001          40        JR   NC,S2
4D1E  15            41        DEC  D
                    42 ;
4D1F  10F3          43 S2:    DJNZ BH              ;test fin colonne
                    44 ;
4D21  EB            45        EX   DE,HL
4D22  015100        46        LD   BC,81
4D25  09            47        ADD  HL,BC           ;colonne suivante
4D26  EB            48        EX   DE,HL
                    49 ;
4D27  C1            50        POP  BC
4D28  10D2          51        DJNZ BCL             ;test fin ecran
4D2A  C9            52        RET
```

```
100 ' ---- Initialisation table adresses rectangles ----
110 '
120 '            SERPENTIN   (gauche->droite)
130 '
```

```
140 MEMORY 20899
150 j=1:MODE 2:cpt=20900
160 '
170 FOR k=1 TO 79 STEP 2
180     FOR i=1 TO 25:GOSUB 270:NEXT i          'haut   -> bas
190     FOR i=25 TO 1 STEP -1:GOSUB 280:NEXT i  'bas    -> haut
200 NEXT k
210 '
220 SAVE "rectab",b,20900,4000
230 END
240 '
250 'calcul adresse ecran du rectangle et mise a jour table
260 '
270 adr=49152+(i-1)*80+k-1:GOTO 290
280 adr=49152+(i-1)*80+k
290 POKE cpt,adr-INT(adr/256)*256
300 POKE cpt+1,INT(adr/256)
310 cpt=cpt+2
320 RETURN
```

```
100 ' ---- Initialisation table adresses rectangles ----
110 '
120 '            SERPENTIN   (BORD->CENTRE)
130 '
140 MEMORY 20899
150 j=1:MODE 2:cpt=20900
160 '
170 FOR k=1 TO 12
180     FOR i=k TO 81-k:GOSUB 340:NEXT i            'gauche -> droite
190     i=i-1
200     FOR j=k+1 TO 26-k:GOSUB 340:NEXT j          'haut   -> bas
210     j=j-1
220     FOR i=80-k TO k STEP -1:GOSUB 340:NEXT i    'droite -> gauche
230     i=i+1
240     FOR j=25-k TO k+1 STEP -1:GOSUB 340:NEXT j  'bas    -> haut
250     j=j+1
260 NEXT k
270 '
280 FOR i=13 TO 68:GOSUB 340:NEXT i                 'fin serpentin
290 '
295 SAVE "rectab",b,20900,4000
300 END
310 '
320 'calcul adresse ecran du rectangle et mise a jour table
330 '
340 adr=49152+i-1+(j-1)*80
```

```
350 POKE cpt,adr-INT(adr/256)*256
360 POKE cpt+1,INT(adr/256)
370 cpt=cpt+2
380 RETURN
```

```
                    1 ; ---- INITIALISATION T3: SERPENTIN BORDS->CENTRE ----
                    2 ;
4D58                3         ORG  19800
4D58  21A451        4         LD   HL,20900          ;debut T3
4D5B  1100C0        5         LD   DE,#C000          ;debut ecran
4D5E  3E01          6         LD   A,1               ;pointeur tours
4D60  064F          7         LD   B,79              ;longueur 1ere ligne
4D62  0EFF          8         LD   C,#FF             ;1-2*A
4D64  F5            9         PUSH AF
4D65  180A         10         JR   ENTREE
                   11 ;
4D67  F5           12 RETOUR: PUSH AF                ;sauvegarde ptr tours
                   13 ;
4D68  07           14         RLCA                   ;2*A
4D69  3D           15         DEC  A                 ;2*A-1
4D6A  ED44         16         NEG                    ;1-2*A
4D6C  4F           17         LD   C,A               ;sauvegarde dans C
                   18 ;
4D6D  3C           19         INC  A                 ;calcul longueur ligne
4D6E  C650         20         ADD  A,80              ;du haut
4D70  47           21         LD   B,A               ;-> dans B
                   22 ;
4D71  73           23 ENTREE: LD   (HL),E            ;chargement T3
4D72  23           24         INC  HL
4D73  72           25         LD   (HL),D
4D74  23           26         INC  HL
                   27 ;
4D75  13           28         INC  DE                ;parcours gauche-droite
4D76  10F9         29         DJNZ ENTREE            ;test fin ligne
                   30 ;
4D78  79           31         LD   A,C               ;calcul hauteur colonne
4D79  C619         32         ADD  A,25              ;de droite
4D7B  47           33         LD   B,A               ; -> dans B
                   34 ;
4D7C  73           35 HB:     LD   (HL),E            ;chargement T3
4D7D  23           36         INC  HL
4D7E  72           37         LD   (HL),D
4D7F  23           38         INC  HL
                   39 ;
4D80  7B           40         LD   A,E               ;parcours haut-bas
4D81  C650         41         ADD  A,80
4D83  5F           42         LD   E,A
```

```
4D84  3001      43        JR   NC,S1
4D86  14        44        INC  D
4D87  10F3      45 S1:    DJNZ HB              ;test fin colonne
                46 ;
4D89  79        47        LD   A,C             ;calcul longueur ligne
4D8A  C650      48        ADD  A,80            ;du bas
4D8C  47        49        LD   B,A             ;-> dans B
                50 ;
4D8D  73        51 GD:    LD   (HL),E          ;chargement T3
4D8E  23        52        INC  HL
4D8F  72        53        LD   (HL),D
4D90  23        54        INC  HL
                55 ;
4D91  1B        56        DEC  DE              ;parcours droite-gauche
4D92  10F9      57        DJNZ GD              ;test fin ligne
                58 ;
4D94  79        59        LD   A,C             ;calcul hauteur colonne
4D95  C618      60        ADD  A,24            ;de gauche
4D97  47        61        LD   B,A             ;-> dans B
                62 ;
4D98  73        63 BH:    LD   (HL),E          ;chargement T3
4D99  23        64        INC  HL
4D9A  72        65        LD   (HL),D
4D9B  23        66        INC  HL
                67 ;
4D9C  7B        68        LD   A,E             ;parcours bas-haut
4D9D  D650      69        SUB  80
4D9F  5F        70        LD   E,A
4DA0  3001      71        JR   NC,S2
4DA2  15        72        DEC  D
4DA3  10F3      73 S2:    DJNZ BH              ;test fin colonne
                74 ;
4DA5  F1        75        POP  AF
4DA6  3C        76        INC  A
4DA7  FE0D      77        CP   13              ;test dernier tour
4DA9  20BC      78        JR   NZ,RETOUR
                79 ;
4DAB  0639      80        LD   B,57            ;dernier tour: 1 ligne
4DAD  73        81 DL:    LD   (HL),E          ;chargement T3
4DAE  23        82        INC  HL
4DAF  72        83        LD   (HL),D
4DB0  23        84        INC  HL
4DB1  13        85        INC  DE              ;parcours gauche-droite
4DB2  10F9      86        DJNZ DL              ;test fin derniere ligne
4DB4  C9        87        RET
```

La myriade (affichage aléatoire)

Le module RECTAN.ASS étant indépendant de la table de coordonnées T3, nous pouvons imaginer d'autres modes d'affichage des rectangles ; on pourrait envisager un affichage aléatoire ; disons plutôt "pseudo-aléatoire" : il faut bien se servir de la table, ce qui évite d'écrire une autre routine assembleur ; nous allons construire une table ordonnée, et procéder à un mélange aléatoire des éléments de cette table ; le résultat est donc imprévisible mais lié à la table.

La construction directe d'une table au hasard pose certains problèmes : en effet, on génère des adresses aléatoirement, et on écrit une adresse en mémoire s'il s'agit bien d'un nouveau couple ; afin d'accélérer le traitement à un certain point de la génération, il devient indispensable de ne plus générer d'adresses (ou d'en générer moins !) afin de ne plus être obligé de balayer le début de la table pour vérifier l'unicité ; si vous ne mettez pas en œuvre un tel algorithme, la génération va se ralentir au fur et à mesure de l'avancement, et il sera impossible de terminer un tel traitement dans un délai raisonnable...

Nous avons donc utilisé un artifice, consistant à construire une première table ordonnée, et à mélanger ensuite les couples au hasard. Le programme RECTAN.ASS exploitera la table ainsi générée.

Voici la liste du **programme Basic RECALEA** :

```
100 ' ---- Initialisation table adresses rectangles ----
110 '
120 '                MYRIADE
130 '
140 MEMORY 20899:MODE 2:DEFINT i-z
150 '
160 DIM z(2000):FOR i=1 TO 2000:z(i)=i:NEXT i
170 '
171 FOR i=2000 TO 2 STEP -1
172   k=INT(RND*i)+1
173   s=z(k):z(k)=z(i):z(i)=s
174 NEXT i
180 FOR i=1 TO 2000
200     a=49151+i
210     POKE (2*z(i)+20898),a-INT(a/256)*256
220     POKE (2*z(i)+20899),INT(a/256)
230 NEXT i
240 SAVE "recalea",b,20900,4000
250 END
260 '
270 'Remarque:
280 '   -49152=adresse debut 1er bloc ecran (bloc des lignes 1,9,17,...,193)
290 '   -51151=adresse fin 1er bloc ecran
295 '     => 2000 adresses: 80 par ligne et 25 lignes
300 '   -Methode : on melange une table ordonnee
```

D'un point de vue pratique, il suffit donc de charger le programme d'animation choisi, de charger également soit les tables générées par les programmes Basic, soit les modules Assembleur de création de ces tables (les programmes peuvent être tous présents en même temps en mémoire), de rappeler l'image (ou les images) à animer et d'effectuer des "CALL" appropriés... A vous de jouer !

Chapitre 3

LES ANIMATIONS "PLEINE PAGE"

Ces animations ne font pas appel à des tables d'adresses ; elles reprennent les principes de symétrie décrits dans le programme FENETR.ASS, mais, dans le cas présent, la symétrie (horizontale ou verticale) est calculée sur la totalité de l'image écran ; l'exécution du calcul sur la page graphique affichée produit une animation particulière.

Voici les listes des deux modules **Assembleur INVVER.ASS** et **INVHOR.ASS** réalisant les symétries respectivement verticale et horizontale :

```
                    1 ;   -------    INVERSION DROITE <-> GAUCHE    -------
                    2 ;
4A9C                3         ORG  19100
4A9C  06C8          4         LD   B,200              ;nombre de lignes
4A9E  2100C0        5         LD   HL,#C000           ;debut 1ere ligne
                    6 ;                                  ->HL=pointeur gauche
4AA1  114FC0        7         LD   DE,#C04F           ;fin 1ere ligne
                    8 ;                                  ->DE=pointeur droite
4AA4  C5            9 HOR1:   PUSH BC
4AA5  E5           10         PUSH HL
4AA6  D5           11         PUSH DE
4AA7  0628         12         LD   B,40               ;nombre de colonnes / 2
```

```
                    13 ;
4AA9  1A            14 HOR2:   LD   A,(DE)
4AAA  CDC54A        15         CALL RENVER              ;inversion de (DE)
4AAD  4F            16         LD   C,A
4AAE  7E            17         LD   A,(HL)
4AAF  CDC54A        18         CALL RENVER              ;inversion de (HL)
4AB2  12            19         LD   (DE),A              ;echange (HL)<->(DE)
4AB3  71            20         LD   (HL),C
4AB4  23            21         INC  HL                  ;pointeur gauche + 1
4AB5  1B            22         DEC  DE                  ;pointeur droite - 1
4AB6  10F1          23         DJNZ HOR2                ;test fin ligne
                    24 ;
4AB8  E1            25         POP  HL
4AB9  CD26BC        26         CALL #BC26               ;pointeur droite:
                    27 ;                                   -> ligne suivante
4ABC  EB            28         EX   DE,HL
4ABD  E1            29         POP  HL
4ABE  CD26BC        30         CALL #BC26               ;pointeur gauche:
                    31 ;                                   -> ligne suivante
4AC1  C1            32         POP  BC
4AC2  10E0          33         DJNZ HOR1
4AC4  C9            34         RET
                    35 ;
                    36 ;  ---  renversement d'un octet image contenu dans A ---
                    37 ;              point gauche <-> point droite
                    38 ;
                    39 ; <=> echange bits 7 et 6, 5 et 4, 3 et 2, 1 et 0.
                    40 ;
4AC5  C5            41 RENVER: PUSH BC
4AC6  47            42         LD   B,A
4AC7  E6AA          43         AND  %10101010           ;masquage point gauche
4AC9  1F            44         RRA                      ;devient point droite
4ACA  4F            45         LD   C,A
4ACB  78            46         LD   A,B
4ACC  E655          47         AND  %01010101           ;masquage point droite
4ACE  17            48         RLA                      ;devient point gauche
4ACF  B1            49         OR   C                   ;reunion des 2 points
4AD0  C1            50         POP  BC
4AD1  C9            51         RET
```

```
                     1 ;--------  INVERSION VERTICALE  --------
                     2 ;
                     3 ;-> inversion haut<->bas de l'ecran
                     4 ;
                     5 ;-> Routines systeme utilisees:
                     6 ;         - BB29 : calcul adresse ligne superieure
                     7 ;         - BC26 : calcul adresse ligne inferieure
```

```
                    8 ;
                    9 ;-> zone de transfert utilisee:
                   10 ;         - adresse : 20020
                   11 ;         - longueur: 80
                   12 ;
4A38               13          ORG  19000
4A38  0664         14          LD   B,100            ;nombre de lignes/2
4A3A  2100C0       15          LD   HL,#C000         ;adresse premiere ligne
                   16 ;                                ->ligne haute
4A3D  1180FF       17          LD   DE,#FF80         ;adresse derniere ligne
                   18 ;                                ->ligne basse
4A40  C5           19 RETOUR:  PUSH BC
4A41  E5           20          PUSH HL
4A42  D5           21          PUSH DE
                   22 ;
4A43  11344E       23          LD   DE,20020         ;adresse zone transfert
4A46  015000       24          LD   BC,80            ;longueur d'une ligne
4A49  EDB0         25          LDIR                  ;transfert lg haute->zone
4A4B  E1           26          POP  HL
4A4C  D1           27          POP  DE
4A4D  D5           28          PUSH DE
4A4E  E5           29          PUSH HL
4A4F  015000       30          LD   BC,80
4A52  EDB0         31          LDIR                  ;lg basse -> lg haute
4A54  21344E       32          LD   HL,20020
4A57  D1           33          POP  DE
4A58  D5           34          PUSH DE
4A59  015000       35          LD   BC,80
4A5C  EDB0         36          LDIR                  ;zone -> lg basse
                   37 ;
4A5E  E1           38          POP  HL
4A5F  CD29BC       39          CALL #BC29            ;montee ligne basse
4A62  EB           40          EX   DE,HL
4A63  E1           41          POP  HL
4A64  CD26BC       42          CALL #BC26            ;descente ligne haute
                   43 ;
4A67  C1           44          POP  BC
4A68  10D6         45          DJNZ RETOUR
4A6A  C9           46          RET
```

L'inversion de couleurs déjà présentée dans le module FENETR.ASS consiste à effectuer le complément à 1 de chaque octet de codification de l'image (instruction Assembleur CPL : complément à 1 du contenu de l'accumulateur) : le module **Assembleur INVCOUL.ASS** effectue cette transformation sur la totalité de la page-écran ; voici la liste :

```
                    1 ; --------  INVERSION DES COULEURS  --------
                    2 ;
4B00                3         ORG  19200
4B00  2100C0        4         LD   HL,#C000          ;debut ecran
4B03  010040        5         LD   BC,#4000          ;longueur ecran
                    6 ;
4B06  7E            7 BCL:    LD   A,(HL)
4B07  2F            8         CPL                    ;inversion
4B08  77            9         LD   (HL),A
4B09  23           10         INC  HL                ;octet suivant
4B0A  0B           11         DEC  BC
4B0B  78           12         LD   A,B
4B0C  B1           13         OR   C
4B0D  20F7         14         JR   NZ,BCL            ;test fin ecran
4B0F  C9           15         RET
```

Des animations spectaculaires peuvent être réalisées en faisant "cycler" les couleurs ; cette méthode, simple à mettre en œuvre, consiste à modifier l'affectation des couleurs d'encre d'une image donnée ; le résultat est dépendant de votre imagination et de la qualité de vos images ; imaginez par exemple le dessin d'une cascade : le cyclage de quelques couleurs composantes de l'eau de la cascade peut reproduire le mouvement de chute de la masse d'eau. On peut envisager de faire disparaître un objet en le faisant tendre progressivement vers la couleur de fond ; encore une fois, les contraintes techniques (palette réduite à 27 couleurs) réduisent le champ d'expérimentation, mais ce principe peut cependant procurer des résultats inattendus.

Chapitre 4

L'ANIMATION DE CINQ PAGES GRAPHIQUES

Avec les modèles CPC 464 ou 664, il est possible de gérer jusqu'à trois pages graphiques en mémoire, sous réserve que les programmes actifs ne soient pas trop longs.

Sur le modèle CPC 6128, il est possible de gérer en Basic jusqu'à cinq images ; le résultat que nous vous proposons sur la disquette est une prise de judo : le mouvement a été décomposé en cinq séquences correspondant chacune à une page graphique.

Sur le 6128, on dispose d'un banc de mémoire supplémentaire de 64K ; contrairement à CP/M, le Basic n'utilise pas la totalité de l'espace disponible ; nous devons pour cela utiliser les ressources du programme "BANKMAN.BAS" (disquette système n° 1), qui permet de gérer cinq blocks de 16K : quatre blocks correspondent au deuxième banc, et un block correspond à la page écran active ; deux commandes sont à notre disposition afin de gérer ces blocks :

- **!SCREENCOPY** transfère une page graphique sur une autre en écrasant le contenu de cette dernière ;
- **!SCREENSWAP** permute le contenu de deux pages graphiques (le temps de traitement est un peu plus long).

Notre animation de judo fait appel à la commande !SCREENCOPY ; afin d'éviter l'affichage peu esthétique d'une image, nous avons utilisé

un artifice consistant à modifier – avant l'affichage – les affectations de couleurs d'encre de l'image (principe du cyclage des couleurs) en donnant la valeur "blanc brillant" à l'ensemble des seize encres, et à restaurer la table des couleurs après l'affichage. Bien que produisant un effet de "flash" à l'écran, le passage d'une séquence à une autre est plus sympathique.

```
10 'Chargement images
20 '
30 GOSUB 270
40 FOR n=2 TO 5
50   fich$="IMAGE"+CHR$(n+64):LOAD fich$
60   |SCREENCOPY,n,1
70 NEXT n
80 LOAD "IMAGEA"
90 MODE 0:GOSUB 210:FOR k=1 TO 2000:NEXT k
100 '
110 'Affichage images
120 '
130 FOR n2=2 TO 5
140    FOR k=1 TO 500:NEXT k
150    GOSUB 270:|SCREENCOPY,1,n2:GOSUB 210
160 NEXT n2
170 END
180 '
190 'Mise a jour encres
200 '
210 RESTORE 230
220 FOR k=0 TO 15:READ c:INK k,c:NEXT k:BORDER 26:RETURN
230 DATA 26,25,10,19,23,14,2,3,10,9,15,16,6,1,22,0
240 '
250 'Mise a blanc des encres 0 a 15
260 '
270 FOR i=0 TO 15:INK i,26:NEXT i:RETURN
```

Conclusion

La création d'images sur ordinateur est bien plus qu'un phénomène de mode, car l'outil informatique a séduit de nombreux graphistes par la richesse des expérimentations possibles ; le micro-ordinateur AMSTRAD constitue un bon outil d'apprentissage.

Le plus délicat est de "dompter" la machine, afin de se forger un style, car l'image est fortement conditionnée par la technologie qui la réduit invariablement en une matrice de pixels (pixel = contraction de "picture elements").

La construction d'animations est parfois contraignante, mais les vraies limites sont celles de notre imagination ; les modules que nous vous avons présentés servent de base de travail à une recherche sur des variations possibles (modification des paramètres, changement d'ordre d'une table d'adresses...) ; à un certain avancement de la programmation, il faut donc procéder par "essais-erreurs" – en modifiant un ou plusieurs paramètres – afin de visualiser de nouveaux modes d'animation.

Nous avons évoqué la possibilité de construire des "bandes dessinées électroniques", appelons-les "micro-clips" ; ce type de produit cadre parfaitement avec les outils techniques dont nous disposons ; la bande dessinée est un moyen d'expression très visuel impliquant une lecture et "consommation" rapide, chaque image étant le support d'autres images que nous inventons nous-mêmes ; les animations sur micro-ordinateur procurent les mêmes sensations d'impact, de vitesse, mais une dizaine d'images, découpées, transformées, semblent cependant limiter le récit : il ne s'agit pas de tout dire, mais de suggérer, notre imagination faisant le reste.

Annexe 1

LE GUIDE UTILISATEUR DE LA DISQUETTE

CRÉATION DE FORMES

Ce programme s'utilise exclusivement au clavier ; après la sélection, un sous-menu vous propose la création d'un objet, la visualisation des objets contenus dans la table TABOBJ, la modification d'un objet existant, la suppression d'un objet du catalogue, et enfin l'enregistrement (sauvegarde) de l'objet que vous venez de créer. Pour accéder à une de ces fonctions, il suffit de taper le numéro correspondant.

Vous pouvez créer des motifs inscrits dans un rectangle de taille maximale 20 par 20 pixels ; lorsque vous sélectionnez l'option création, le programme vous demande la taille du rectangle nécessaire à la fabrication, puis vous affiche un écran résumant les commandes disponibles ("HELP"); voici la liste de ces commandes :

"flèches"	: déplacement du curseur ;
f8	: coloriage vertical vers le haut ;
f2	: coloriage vertical vers le bas ;
f4	: coloriage du début de la ligne ;
f6	: coloriage de la fin de ligne ;
"RETURN"	: coloriage de la case ;
F	: fin du dessin ;

C : choix d'une couleur sur la palette standard les flèches gauche et droite positionnent une flèche de repérage de la couleur en cours sur la palette.

Après avoir lu les commandes, vous appuyez sur une touche pour passer en mode fabrication ; une grille apparaît alors, correspondant à la taille du rectangle de travail choisi ; la case active est repérée par un curseur ovale ; la palette de couleur est inscrite en bas de l'écran. A vous de jouer pour créer un motif.

A tout moment de la fabrication, vous pouvez visualiser votre objet en taille réelle (un rectangle de la grille = un pixel), en tapant "F" puis "2" ; en mode visualisation, tapez sur une touche pour revenir au sous-menu, et tapez "3" pour poursuivre votre construction (vous êtes alors en mode modification).

Pour supprimer un travail, il faut donner le numéro de l'objet correspondant ; la phase de sauvegarde mettra à jour la table d'objets, l'ensemble des actions (création, modification ou éventuellement suppression) étant alors enregistré sur la disquette.

DÉPLACEMENT DE FORMES (JOYSTICK OU SOURIS)

Avant de choisir une des cinq méthodes exposées dans le chapitre 3 (méthodes de l'empreinte XOR1 et XOR2, méthode du déplacement du rectangle-objet, déplacement de la forme inscrite dans le rectangle, ou déplacement d'un caractère), vous pouvez régler la vitesse de déplacement de l'objet en choisissant les pas de déplacement suivant les axes X et Y, et en spécifiant la présence ou non d'une synchronisation "déplacement/affichage".

Ce programme et les suivants sur cette face de la disquette fonctionnent au joystick, ou avec une souris ; pour revenir à un sous-menu ou au menu général, vous devez taper "ESC".

CRAYON

Cliquez pour activer le mode "tracé", et déplacez le crayon à l'écran pour dessiner ; cliquez à nouveau pour sortir du mode "tracé" et revenir au mode "déplacement sans tracé".

GOMME

Le fonctionnement est similaire ; la gomme exécute un tracé dans la couleur de fond. Nous avons chargé une image afin de matérialiser le résultat de l'effacement.

AÉROGRAPHE

Cliquez pour activer le mode "aérographe" ; en déplaçant le rectangle de repérage, vous répandez des points de couleur à l'écran ; les points sont émis aléatoirement à l'intérieur du petit rectangle lors du déplacement ; cliquez à nouveau pour quitter le mode "aérographe".

FENÊTRE

Après le chargement de "la lanceuse de poids", vous pouvez déplacer une flèche de repérage ; positionnez la flèche et cliquez, déplacez-vous alors vers le bas et vers la droite de l'écran par exemple afin de délimiter une fenêtre de travail ; lorsque votre fenêtre est satisfaisante, cliquez à nouveau. Vous pouvez alors effectuer dans cette zone :

- une symétrie horizontale en tapant "H" ;
- une symétrie verticale en tapant "V" ;
- ou une inversion de couleur en tapant "C".

Vous pouvez, bien sûr, combiner ces trois commandes ; en cliquant à nouveau, le cadre de travail disparaît (les transformations sont conservées), et la flèche de repérage est à nouveau disponible pour la définition d'un nouveau cadre.

Après être sorti du programme (sélectionner "FIN" sur le menu principal), vous pouvez retourner la disquette et accéder aux animations en tapant également RUN "MENU. Cette face correspond au deuxième chapitre du livre. Le menu principal apparaît en vous proposant trois familles d'animation, et l'animation de cinq images. Pour chaque famille sélectionnée sera présenté un sous-menu de variantes.

SCROLLING

Les quatre options suivantes sont proposées :

- descendant, image droite ;
- descendant, image inversée ;
- montant, image droite ;
- ou montant, image inversée.

AFFICHAGE DE RECTANGLES

Quatre options sont également présentées :

- myriade (affichage aléatoire des 2 000 rectangles) ;
- affichage colonne (affichage ordonné, colonne par colonne) ; au sein d'une colonne, l'affichage est alternativement montant et descendant ; il est nécessaire de temporiser l'animation si l'on veut percevoir la totalité du mouvement ;
- affichage ligne ;
- serpentin.

Dans chaque option, l'animation est double, animation sur une image, puis retour à un écran vierge selon le même principe.

SYMÉTRIES

Une image initiale est chargée, et une symétrie horizontale ou verticale est alors calculée sur cette image.

ANIMATION DE CINQ IMAGES

Cette animation ne fonctionne que sur le modèle CPC 6128, elle présente une prise de judo décomposée en cinq étapes ; les différentes pages graphiques sont affichées successivement (c'est la seule animation programmée en Basic).

Annexe 2

LA DESCRIPTION DES CATALOGUES DE LA DISQUETTE

CATALOGUE DISQUETTE FACE A

Drive A: user 0

AERO	,,ASS	1K	MENU	,,BAS	3K
AERO	,,TXT	4K	POIDS	,,BIN	17K
AFFCAD	,,ASS	1K	SAUT	,,BIN	17K
AFFCAD	,,TXT	1K	TABOBJ	,,BIN	1K
AFFCRE	,,ASS	1K	TRACE	,,ASS	1K
AFFCRE	,,TXT	2K	TRACE	,,TXT	2K
CREOBJ	,,BAS	6K	TRACE1	,,ASS	1K
DEPLC	,,ASS	1K	TRACE1	,,TXT	1K
DEPLC	,,TXT	1K	TRACE2	,,ASS	1K
DEPLF	,,ASS	1K	TRACE2	,,TXT	2K
DEPLF	,,TXT	1K	VALID	,,ASS	1K
DEPLZ	,,ASS	1K	VALID	,,TXT	2K
DEPLZ	,,TXT	1K	XOR1	,,ASS	1K
FENETR	,,ASS	1K	XOR1	,,TXT	1K
FENETR	,,TXT	6K	XOR2	,,ASS	1K
GOMME	,,ASS	1K	XOR2	,,TXT	2K
GOMME	,,TXT	2K	ZOOM	,,ASS	1K
HAIE	,,BIN	17K	ZOOM	,,TXT	2K
JAVELOT	,,BIN	17K			

55K free

CATALOGUE DISQUETTE FACE B

Drive A: user 0

BOXE	,,BIN	17K	RECALEA	,,BIN	5K
IMAGEA	,,BIN	17K	RECGADR	,,BAS	1K
IMAGEB	,,BIN	17K	RECGD	,,ASS	1K
IMAGEC	,,BIN	17K	RECGD	,,TXT	1K
IMAGED	,,BIN	17K	RECHB	,,ASS	1K
IMAGEE	,,BIN	17K	RECHB	,,TXT	1K
INT1T2	,,ASS	1K	RECHOBA	,,BAS	1K
INT1T2	,,TXT	1K	RECSER	,,BAS	1K
INVCOUL	,,ASS	1K	RECTAB	,,BIN	5K
INVCOUL	,,TXT	1K	RECTAN	,,ASS	1K
INVHOR	,,ASS	1K	RECTAN	,,TXT	2K
INVHOR	,,TXT	2K	RECTIN	,,ASS	1K
INVTAB	,,ASS	1K	RECTIN	,,TXT	2K
INVTAB	,,TXT	1K	SCROLL	,,ASS	1K
INVVER	,,ASS	1K	SCROLL	,,TXT	1K
INVVER	,,TXT	1K	TSCROLL	,,BAS	1K
MENU	,,BAS	3K	TSCROLL	,,BIN	1K
RECALEA	,,BAS	1K			

34K free

Annexe 3

LA LISTE DES ROUTINES "ROM" UTILISEES

GESTION DU CLAVIER ET DU JOYSTICK

☐ **BB1B** : teste si une touche quelconque est appuyée.

Entrée : rien.
Sortie : si une touche est appuyée alors le registre CARRY est à 1,
A contient le caractère frappé ;
les registres sont inchangés.

☐ **BB1E** : teste si une touche précise est appuyée.

Entrée : A contient la valeur d'une touche
Sortie : si la touche n'est pas appuyée alors Z =1.
sinon Z=0, A et HL sont modifiés.

☐ **BB24** : lit le joystick.

Entrée : rien.
Sortie : A et H contiennent l'état du joystick "0", L contient l'état du joystick "1" (la valeur d'état est celle renvoyée par l'instruction JOY en Basic).

GESTION GRAPHIQUE

☐ **BBC0** : déplace le curseur graphique en coordonnées absolues.

Entrée : DE = abscisse X.
HL = ordonnée Y.
Sortie : registres modifiés.

☐ **BBC3** : déplace le curseur graphique en coordonnées relatives.

Entrée : DE = abscisse X.
HL = ordonnée Y.
Sortie : registres modifiés.

☐ **BBC6** : lit la position du curseur graphique.

Entrée : rien.
Sortie : DE = abscisse X.
HL = ordonnée Y.

☐ **BBDE** : positionne le stylo graphique.

Entrée : A = encre (0 à 15).
Sortie : AF modifié.

☐ **BBF6** : trace une ligne en coordonnées absolues depuis la position graphique courante.

Entrée : DE = abscisse X du point d'arrivée.
HL = ordonnée Y du point.
Sortie : AF,BC,HL et DE modifiés.

☐ **BBF9** : trace une ligne en coordonnées relatives depuis la position graphique courante.

Entrée : DE = déplacement X.
HL = déplacement Y.
Sortie : AF,BC,HL et DE modifiés.

☐ **BBFC** : écrit un caractère à la position graphique courante.

Entrée : A = code du caractère.
Sortie : AF, BC, HL et DE modifiés le curseur graphique est déplacé de 8 points vers la droite.

☐ **BC1D** : calcule l'adresse d'un point.

Entrée : DE = abscisse X.
HL = ordonnée Y.
(attention : coordonnées réelles variant suivant le mode).

Sortie : HL = adresse réelle du point.
B = nombre de points par octets − 1.
C = masque pour le point.
DE et AF modifiés.

☐ **BC26** : calcule l'adresse écran de l'octet situé sur la ligne inférieure.

Entrée : HL contient l'adresse écran actuelle.
Sortie : HL contient la nouvelle adresse et AF est modifié.

☐ **BC29** : calcule l'adresse écran de l'octet situé sur la ligne supérieure.

Entrée : HL contient l'adresse écran actuelle.
Sortie : HL contient la nouvelle adresse, et AF est modifié.

☐ **BC2C** : transformation d'une encre (numéro de stylo) en un masque représentant un octet écran dont tous les points sont de la couleur de l'encre.

Entrée : A = encre (0 à 15).
Sortie : A = masque écran.

☐ **BC2F** : opération inverse de la routine BC2C.

☐ **BC59** : établit le mode d'écriture des graphiques.

Entrée : A = mode soit 0 = normal
1 = OU exclusif
2 = ET
3 = OU
Sortie : AF, BC, HL et DE modifiés.

☐ **BD19** : attend une interruption correspondant au début de balayage vertical.

DISQUETTE D'ACCOMPAGNEMENT

Cette disquette est le complément du livre. Tous les programmes décrits dans le livre y sont reproduits.

Comment utiliser cette disquette ?

Après avoir mis votre micro-ordinateur AMSTRAD sous tension, vous êtes automatiquement sous BASIC (prompt "Ready"); insérez la face "A" de la disquette et tapez RUN "MENU puis RETURN. Le Menu "CREATION GRAPHIQUE" apparaît alors, il correspond au premier chapitre du livre. La sélection d'une fonction de dessin ou de transformation d'une image se fait en déplaçant les flèches "haut" et "bas", puis en tapant sur "RETURN".

N'oubliez pas de sauvegarder vos données en modifiant le nom du fichier. Vous pourrez ainsi récupérer les fichiers originaux si besoin est.

Vente par correspondance

BON DE COMMANDE

Je commande la disquette d'accompagnement du livre **"CRÉATION ET ANIMATIONS GRAPHIQUES SUR AMSTRAD"** au prix de 150,00 FF* plus 10,00 FF** pour frais de port et d'emballage.

* prix valable jusqu'au 31 décembre 1986
** hors CEE, prévoyez 15,00 FF pour supplément de frais d'expédition.

NOM ..... **Prénom** .....
Adresse .....
.....
..... **Ville** .....
Code Postal ..... **Pays** .....

Renvoyez-nous ce bon rempli (découpé ou copié), avec votre règlement par chèque bancaire ou postal, établi à l'ordre de LA CONSOLE.

Conseils de lecture

Pour approfondir vos connaissances en Basic, mieux connaître le système des CPC 464, 664 et 6128, et maîtriser le graphisme sur Amstrad, P.S.I. vous propose une palette d'ouvrages utiles.

POUR MAITRISER LE BASIC AMSTRAD

☐ **Basic Amstrad 1. Méthodes pratiques** – Jacques Boisgontier et Bruno Césard (Editions du P.S.I.).

Pour ceux qui ont déjà pratiqué un Basic, voici un ouvrage de perfectionnement au Basic Amstrad. Un chapitre sur le CP/M 2.2 et le CP/M Plus donne les principales commandes systèmes.

☐ **Basic Amstrad 2. Programmes et fichiers** – Jacques Boisgontier (Editions du P.S.I.).

Pour pratiquer le Basic Amstrad, cet ouvrage donne de nombreux programmes de gestion, d'éducation et de jeu où le rôle des fichiers est expliqué et largement commenté.

☐ **Basic plus. 80 routines sur Amstrad** – Michel Martin (Editions du P.S.I.).

Pour pousser votre Amstrad au maximum de ses capacités : 80 routines de simulation d'instructions qui n'existent pas en Basic Amstrad.

POUR MIEUX CONNAITRE LE SYSTÈME DES CPC

☐ **Clefs pour Amstrad 1. Système de base** – Daniel Martin (Editions du P.S.I.).

Mémento présentant synthétiquement le jeu d'instructions du Z-80, les points d'entrée des routines systèmes, les connecteurs et brochages, etc.

Le livre de chevet du programmeur sur Amstrad.

☐ **Clefs pour Amstrad 2. Système disque** – Daniel Martin et Philippe Jadoul (Editions du P.S.I.).

Ce deuxième tome consacré au système disque présente les points d'entrée des routines disque, les blocs de contrôle, la programmation et les brochages des circuits spécialisés...

La deuxième partie du livre est aussi destinée aux possesseurs d'Amstrad 8256.

☐ **CP/M plus sur Amstrad** – Yvon Dargery (Editions du P.S.I.).

Toutes les commandes CP/M et CP/M plus pour maîtriser le système des 6128 et 8256 : un ouvrage de référence illustré par de nombreux programmes.

☐ **Le livre de l'Amstrad – Tome 1** – Daniel Martin et Philippe Jadoul (BCM – diffusé par P.S.I.).

Ce livre, destiné aux programmeurs des CPC 464 et 664, donne une étude complète de tous les circuits internes, et analyse la structure interne du BASIC. Vous y trouverez, en outre, une étude complète des RSX, et des programmes de scrolling, de traçage de rectangles, de coloriage de surface et de manipulation vectorielle.

POUR CONCEVOIR ET AMÉLIORER VOS GRAPHISMES

☐ **Mathématiques et graphismes** – Gérald Grandpierre et Richard Cotté (Editions du P.S.I.).

De très beaux graphismes sont générés par des équations mathématiques. L'univers des fractals, les déformations et les enveloppes, les surfaces en Z2 sont étudiées dans ce livre très pédagogique et de haut niveau. Tous les programmes, écrits en Basic standard, sont facilement adaptables au Basic Amstrad.

☐ **Graphisme en Assembleur sur Amstrad CPC** – Francis Piérot (Editions du P.S.I.).

Cet ouvrage met en pratique les notions d'assembleur acquises par le lecteur grâce à de nombreux programmes commentés. Du simple dessin d'un cercle à l'animation d'une soucoupe volante dans un décor en passant par la mise en mouvement d'une corne d'abondance, vous apprendrez à maîtriser l'assembleur et à créer de très belles pages-écran.

Achevé d'imprimer en avril 1986
sur les presses de l'imprimerie Laballery
58500 Clamecy
Dépôt légal : avril 1986
N° d'impression : 604052
N° d'édition : 86595-338-1
ISBN : 2-86595-338-6

Votre avis nous intéresse

Pour nous permettre de faire de meilleurs livres, adressez-nous vos critiques sur le présent ouvrage.

— Ce livre vous donne-t-il toute satisfaction ?

.....

.....

.....

.....

.....

— Y a-t-il un aspect du problème que vous auriez aimé voir abordé ?

.....

.....

.....

.....

.....

Si vous souhaitez des éclaircissements techniques, écrivez-nous, nous ne manquerons pas de vous répondre directement.

Où avez-vous acheté ce livre ?

☐ cadeau ☐ librairie ☐ autres
☐ exposition ☐ boutique micro

Comment en avez-vous eu connaissance ?

☐ publicité ☐ catalogue ☐ autres
☐ exposition ☐ conseils d'un ami

Avez-vous déjà acquis des livres P.S.I. ?

Lesquels ? .....

.....

qu'en pensez-vous ? .....

.....

.....

Nom ..... Prénom ..... Age.....
Adresse .....
Profession .....
Centre d'intérêt .....

P.S.I.
MICRO ET TECHNIQUES

CRÉATION ET ANIMATIONS GRAPHIQUES SUR AMSTRAD CPC

Cet ouvrage vous propose de créer des images à l'aide de la souris et de la manette de jeu, de les animer et de réaliser vos propres "bandes dessinées électroniques" sur un Amstrad 464, 664 ou 6128 (128 K nécessaires pour certaines animations).

Les programmes de dessin et d'animation sont écrits en Basic Amstrad et en assembleur Z80. Ils peuvent constituer une excellente initiation pratique à ces langages.

Vous apprendrez à créer un pinceau, un aérographe, une gomme, puis vous animerez vos dessins par des scrollings, des inversions ou des reconstitutions d'image point par point. Les images couleur contenues dans le livre vous donneront une bonne idée de ce qu'il est possible de réaliser sur le thème du sport.

Gilles Fouchard, informaticien, et Jean-Yves Corre, peintre, vous invitent à pousser à fond les capacités graphiques de votre Amstrad CPC.

9 782865 953387

ÉDITIONS DU P.S.I.
BP 86 - 77402 LAGNY S/MARNE CEDEX - FRANCE

ISBN 2-86595-338-6 110 FF

ILLUSTRATION JEAN-YVES CORRE

Document numérisé avec amour par

AMSTRAD CPC

MÉMOIRE ÉCRITE

https://acpc.me/