AMSTRAD

JEU D'AVENTURE

JEAN-LUC JOUD / CLAUDE VIVIER

SYBEX

AMSTRAD

JEU D'AVENTURE

DANS LA MEME COLLECTION

Amstrad jeux d'action, P. Monsaut
Amstrad premiers programmes, R. Zaks
Amstrad 56 programmes, S.R. Trost
Amstrad guide du BASIC et de l'AMSDOS, J.-L. Gréco/M. Laurent
Amstrad exploré, J. Braga
Amstrad programmation en assembleur, G. Fagot-Barraly
Amstrad guide du graphisme, J. Wynford
Amstrad CP/M 2.2, A. d'Hardancourt
Amstrad astrologie, numérologie, biorythmes, P. Bourgault
Amstrad graphisme en trois dimensions, T. Lachand-Robert
Amstrad Multiplan, Amstrad
Amstrad CP/M plus, A. d'Hardancourt
Amstrad Astrocalc, G. Blanc/P. Destrebecq
Amstrad gagnez aux courses, J.-C. Despoine
Amstrad créer de nouvelles instructions, J.-C. Despoine
Amstrad Locoscript, B. Le Dû
Amstrad mise au point des programmes BASIC, C. Vivier/Y. Jacob
Amstrad jeux en assembleur, E. Ravis
Amstrad techniques de programmation des jeux, G. Fagot-Barraly
Amstrad routines en assembleur, J.-C. Despoine
Amstrad mieux programmer en assembleur, T. Lachand-Robert
Amstrad jeux de réflexion, G. Fagot-Barraly
Amstrad programmes en langage machine, S. Webb
Amstrad PCW 8256/8512 guide du BASIC et de Jetsam, J.-L. Gréco/M. Laurent
Amstrad programmez votre traitement de texte, J.-C. Despoine
Amstrad gérez votre portefeuille boursier, J.-C. Despoine, F. Pierre
Amstrad guide de Logo, A. d'Hardancourt
Amstrad Startext (logiciel de traitement de texte)
Amstrad PC 1512 guide de l'utilisateur, N. Saumont/M. Laurent
Amstrad PC 1512 guide de BASIC 2, M. Laurent/J.-L. Gréco
Amstrad PC 1512 guide de DOS plus, M. Laurent/J.-L. Gréco

CLAUDE VIVIER
JEAN-LUC JOUD

AMSTRAD

JEU D'AVENTURE

Paris • Alameda • Düsseldorf

ISBN 2-7361-0204-5

SOMMAIRE

1. QU'EST-CE QU'UN JEU D'AVENTURE ?

■ LES DIVERS TYPES DE JEUX

Les jeux proposés pour les ordinateurs peuvent essentiellement être classés en trois catégories :

Les jeux d'arcade

Ce sont des jeux d'adresse axés sur le réflexe et dans lesquels un poignet souple et un joystick bien huilé priment sur la réflexion (Casse-brique, Tennis, Star invader, Pack man, etc.).

La qualité principale de ce type de jeu étant la rapidité d'exécution, il peut difficilement tourner sous BASIC pur. Le langage privilégié à utiliser pour le programmer reste donc l'assembleur.

Les jeux d'aventure et de rôle

Le joueur incarne un personnage auquel il est confié une mission. Il doit parcourir un chemin plus ou moins complexe, peuplé d'embûches, et découvrir certains objets ou effectuer certaines actions qui lui permettront de poursuivre sa route avec le maximum de sécurité jusqu'au but qui lui a été fixé. De plus, si tout au long de la partie le joueur doit créer avec intelligence et d'une manière judicieuse son propre environnement physique, intellectuel et matériel, cela devient un jeu de rôle (Donjons et Dragons, etc.).

Pour ce type de jeu, il n'est pas nécessaire d'obtenir une rapidité d'exécution aussi grande que pour les jeux d'arcade. Ils peut donc être programmé et exécuté sous BASIC. Cependant, certaines parties du programme devront être organisées afin d'obtenir une rapidité d'exécution compatible avec la patience ergonomique humaine.

Les jeux de stratégie

Morpion, dames, échecs, etc. Malgré une programmation en assembleur quasi obligatoire, le temps de réflexion de l'ordinateur (nécessaire avant qu'il puisse jouer son coup) est parfois très long, surtout s'il doit prévoir plusieurs coups à l'avance. Contrairement à ce que l'on pourrait penser, ce temps mort (s'il n'est pas exagéré) n'est pas toujours un handicap, car il peut simuler le temps de réflexion d'un adversaire réel et permet au joueur humain de faire l'inventaire des coups possibles à venir et de préparer leurs parades.

■ CONSTITUTION D'UN JEU D'AVENTURE

Un jeu d'aventure est principalement constitué de deux parties : le jeu en lui-même, avec son scénario, son plan et ses actions, et une partie communication homme-ordinateur appelée analyse de syntaxe.

Le scénario

C'est dans cette partie que toute l'imagination délirante et la faconde débordante du programmeur pourront enfin assouvir une passion d'enfance (et néanmoins refoulée) de scénariste. Cela peut aller du preux chevalier devant parcourir les souterrains d'un château médiéval pour aller s'enrichir d'un trésor ou délivrer une gente damoiselle (selon la sensibilité du moment) au valeureux astronaute devant découvrir dans un recoin perdu d'univers le schtroumpfonium vital pour l'avenir de sa planète.

Sauf si vous êtes le super-dieu de la littérature, et de surcroît amoureux de Calliope et de Polymnie, donc capable de redonner du brillant au moindre poncif, nous vous déconseillons d'utiliser les thèmes évoqués ci-dessus : ils ont été largement pillés par les centaines d'auteurs de milliers de jeux d'aventure aux temps héroïques du début de l'informatique familiale.

Plus simplement, pour ce jeu, nous vous proposons de choisir une histoire inspirée des *Trois Mousquetaires* et des ferrets de la reine.

Le terrain de jeu

Il s'agit du plan de l'endroit où évolueront les personnages. Il est constitué d'une succession de lieux précis reliés entre eux par des "chemins". Bien entendu, on ne peut passer d'un lieu à un autre que si l'accès est autorisé (chemin existant et, éventuellement, accessible sous certaines conditions).

Pour des raisons d'intérêt du jeu, de programmation et surtout de place mémoire, il est conseillé de ne pas réaliser un plan trop compliqué. Compte tenu de la capacité mémoire limitée des ordinateurs, le plan que nous vous proposons plus loin nous semble d'une taille maximale pour un jeu classique.

De toute façon, et d'une manière générale, compte tenu de la place mémoire disponible, complexité du plan et nombre d'actions sont inversement proportionnels : un plan simple autorisera un grand nombre d'actions et inversement. A la limite, dans le premier cas, l'unique intérêt du jeu sera de trouver la sortie du labyrinthe (jeu de réflexion),

tandis que dans le second cas, on se rapprochera d'un jeu de rôle. C'est donc, avant toute chose, une question de choix.

Les actions

Il s'agit du scénario détaillé et exhaustif de toutes les actions et de tous les cas possibles, qu'ils soient autorisés ou non, concernant tous les lieux définis dans le plan.

Là encore, il faut éviter le piège de la multiplicité trop importante d'actions complexes, hermétiques ou ésotériques. En effet, si, pour le concepteur, la solution paraît évidente, c'est parce qu'il connaît sur le bout des doigts, pour l'avoir créé et constamment modifié, ce qu'il faut faire et ne pas faire pour arriver au but. Le joueur, lui, ignore tout, y compris le plan. Si vous ne voulez pas que votre œuvre finisse aux oubliettes avant que quelqu'un ait trouvé la solution, il faut constamment avoir à l'esprit, lors de la conception, qu'un jeu d'aventure (ou de rôle) est et doit rester avant toute chose un jeu de réflexion et d'intelligence et non pas une devinette à tiroirs complètement insoluble.

Il est évident que le scénario, le plan du terrain de jeu et la liste des actions sont intimement liés les uns aux autres. Ils seront donc inventés, étudiés et réalisés tout à fait conjointement.

L'analyse de syntaxe

Pour le joueur, c'est le moyen de communiquer ses intentions d'action à l'ordinateur. C'est aussi, pour la machine, le moyen de comprendre l'intention du joueur afin de réaliser l'action du joueur.

La procédure de décryptage d'une phrase est pratiquement identique à celle employée inconsciemment par le cerveau humain : tous les mots d'une phrase à analyser sont comparés, l'un après l'autre, aux mots d'un dictionnaire déjà existant en mémoire, jusqu'à l'obtention d'une occurrence. Si elle se produit, une action est décidée en fonction de la définition de ce mot. Si aucun mot similaire n'existe dans le dictionnaire, la phrase restera obscure et aucune action ne pourra être effectuée.

Plus le dictionnaire est important, plus les possibilités de dialogue homme-machine seront riches, donc intéressantes. Cette possibilité sera toutefois réduite par la place mémoire disponible pour enregistrer un tel dictionnaire.

Bien entendu, une analyse de syntaxe peut être plus ou moins complexe : cela peut aller de l'acceptation d'un mot unique pour la

réponse du joueur à un système expert avec recherche de synonymes, correction automatique des fautes d'orthographe ou encore mise à jour du dictionnaire par l'ordinateur. De nos jours, pour qu'un jeu d'aventure reste compétitif, l'analyse de syntaxe doit être suffisamment évoluée pour permettre d'entretenir un dialogue proche de la réalité, sans imposer trop de contraintes. Enfin, la recherche d'occurrences peut se faire par balayage systématique du dictionnaire, recherche dichotomique, etc. Là encore, il sera question d'encombrement mémoire.

2. LE CADRE DU JEU

■ LE SCÉNARIO

Il s'agit donc d'une histoire inspirée des *Trois Mousquetaires*. "Il était une fois" une charmante reine qui était aimée d'un *beautiful duke !* Afin de lui faire comprendre qu'il ne lui était pas tout à fait indifférent, cette reine lui a offert ses ferrets qu'il a amoureusement rapportés chez lui, là-bas, en Angleterre. Cette belle histoire aurait pu s'arrêter là, mais voilà-t-il pas qu'un grand jaloux de cardinal, prévenu par son espionne préférée, ourdit un sombre complot afin de confondre l'infidèle. Il cafarda vilainement au royal mari qui demanda à sa non moins royale dulcinée de lui montrer les fameux bijoux. Un tantinet gênée, elle lui répondit que puisqu'il en était ainsi, elle les porterait lors du prochain bal de la cour. Ils se séparèrent donc sur ce compromis. La reine fit immédiatement appel à ses mousquetaires bien aimés et les envoya en mission secrète à Londres afin de lui ramener les précieux ferrets. Bien entendu, ces joyeux drilles menèrent à bien cette délicate mission et l'honneur fut sauf !

Tel est le résumé de la "véritable" histoire rapportée par Alexandre Dumas. Tout d'abord, nous ne conserverons de cette anecdote que l'essentiel : d'Artagnan est chargé par la reine d'aller récupérer les ferrets au palais de Buckingham et de les lui rapporter dans un délai que nous fixerons à trois jours.

A partir de là, nous allons nous permettre quelques incartades. Au début du jeu, d'Artagnan est seul dans sa chambre et il ne sait rien de ce qui l'attend. Il doit donc trouver et lire une convocation, obtenir de l'argent, prendre un cheval, inviter ses amis à le suivre et aller à Versailles rencontrer la reine. Celle-ci lui donnera alors le véritable objet de sa mission. Les quatre amis vont donc devoir traverser la Manche, aller à Londres et rencontrer le duc. Lorsqu'ils l'auront convaincu de leur remettre les ferrets, ils n'auront plus qu'à refaire le chemin en sens inverse et rapporter les précieux objets à leur ancien propriétaire.

En lisant ce scénario, tout a l'air enfantin. En fait, comme nous le verrons en détail dans le chapitre donnant la description des actions, nous allons glisser pas mal de graviers dans les sabots des étalons et moult peaux de bananes sous les cothurnes de leurs cavaliers.

■ LE TERRAIN DE JEU

Le terrain de jeu est composé de deux grandes parties : les lieux situés en France et ceux situés en Angleterre, séparés par le trajet en

mer. Le plan de ce terrain est présenté en Figure 1 pour la partie française et en Figure 2 pour la partie anglaise.

LES LIEUX

Il est prudent, si l'on veut éviter d'avoir à le modifier sans cesse, de ne pas faire un plan sans penser aux actions générales qui seront à effectuer dans chaque lieu, ou, au contraire, sans définir un lieu en fonction d'une action originale. Le plan du terrain de jeu se fera donc en imaginant déjà des actions ou des conditions rudimentaires.

Les pièges seront de deux sortes : ceux qui ne feront perdre que du temps et ceux qui seront fatals. On pourra même scinder la seconde catégorie en trois : les pièges fatals sans condition préalable et à effet immédiat, ceux à effet retardé, donc avec condition préalable, et enfin les pièges fatals sans effet apparent, qui autoriseront la poursuite du jeu sans pouvoir cependant conclure.

Les actions ou conditions seront précisées en détail dans le chapitre Actions. Pour l'instant, et d'une manière générale, nous avons imaginé pour vous les lieux suivants :

La chambre de d'Artagnan (17)

C'est le point de départ. Notre héros y trouvera d'abord la description de la première partie de sa mission, puis les premiers objets nécessaires à son accomplissement. Ne pas posséder ces objets n'amènera rien dans l'immédiat, mais constituera un piège fatal à effet retardé.

Rue de Trousse-Chemise (1)

Elle sert de carrefour dès la sortie de la chambre. On pourra y mettre des créanciers pour empêcher d'Artagnan de sortir ou d'entrer normalement dans sa chambre.

L'écurie (21)

Étant donné qu'à cette époque on ne se déplaçait qu'à cheval, il sera obligatoire, si l'on veut arriver à temps, de s'en procurer un parmi plusieurs. Bien entendu, cela fera l'objet de pièges fatals à effet retardé.

L'échoppe de l'usurier (22)

Même à cette époque, on n'a rien gratuitement. Nous imaginerons donc un système de paiement pour des services ou des objets utiles

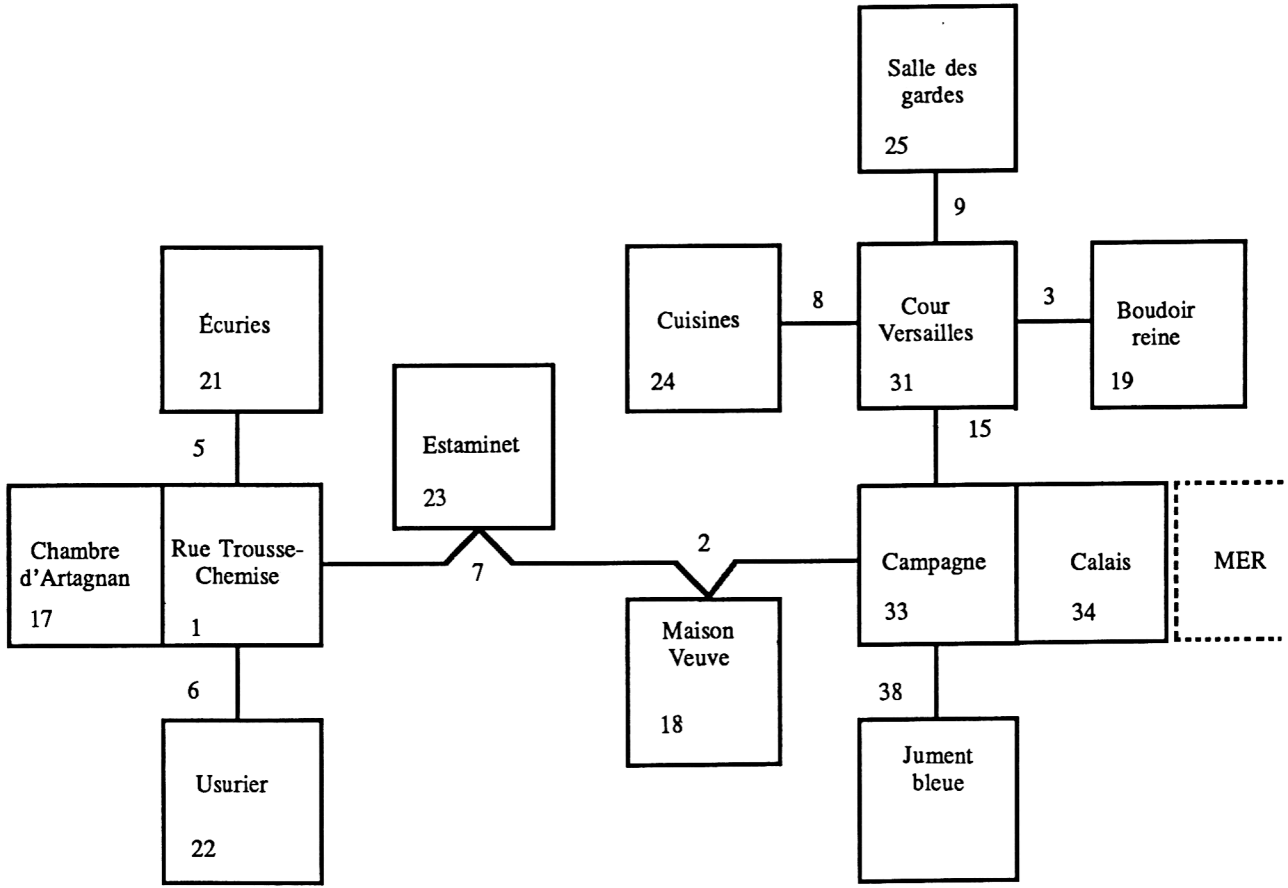


Figure 1 : Plan du jeu (partie française).

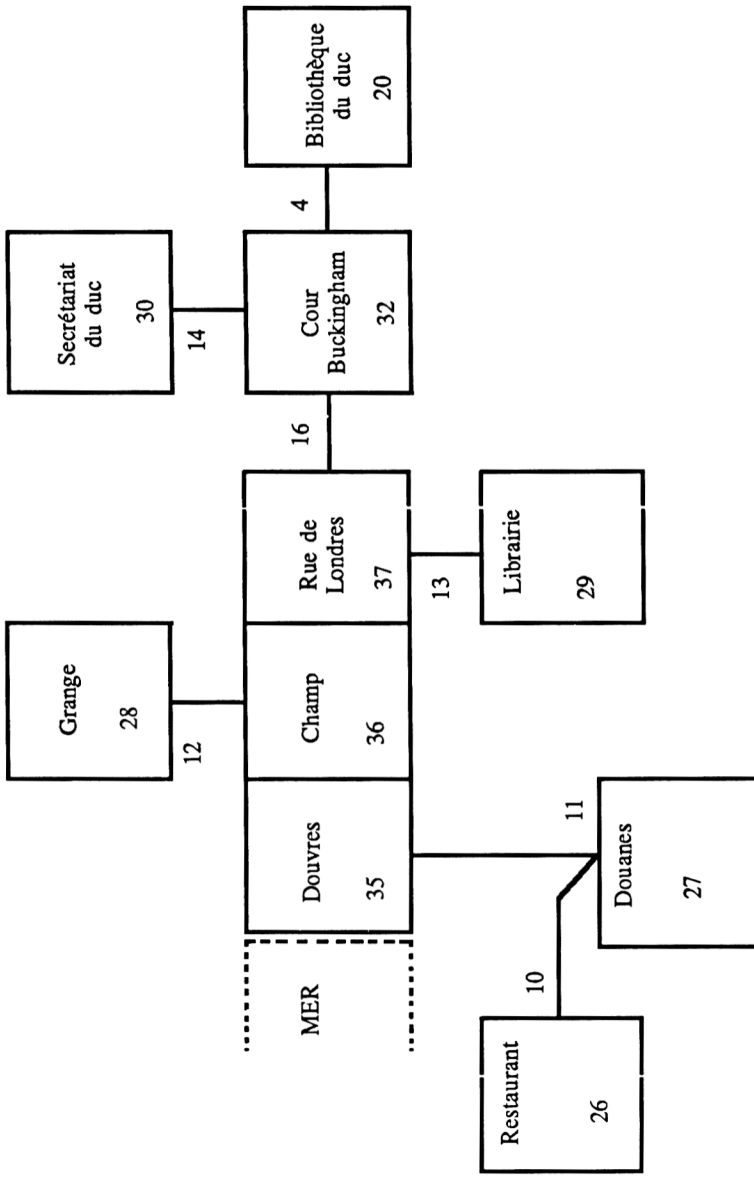


Figure 2 : Plan du jeu (partie anglaise).

et surtout inutiles. L'échoppe de l'usurier a donc été créée afin d'obtenir juste les subsides nécessaires. Ce sera le seul lieu du jeu dans lequel on pourra avoir de l'argent. Il faudra penser aux pistoles françaises, mais aussi aux livres anglaises.

L'estaminet (23)

C'est là que d'Artagnan retrouvera ses amis et peut-être aussi, pourquoi pas, quelques gardes du cardinal...

La maison de la veuve (18)

Bien tentante, mais elle ne servira à rien ! C'est un cul-de-sac. Il faut bien tromper le joueur ! Elle fera perdre du temps si l'on y entre. On pourra même y prévoir un piège fatal immédiat sans condition préalable.

La campagne (33)

Elle sert de carrefour. Généralement on y trouve des fleurs. Qui dit fleurs dit bouquet et aussi une dame à qui les offrir pour obtenir ses grâces... (voir secrétariat du duc). On prévoira trois endroits dans lesquels on pourra se procurer ces fleurs et un seul sera le bon.

La cour de Versailles (31)

Elle sert aussi de carrefour. On pourra y mettre une condition pour aller vers le boudoir de la reine.

Les cuisines (24)

Il s'agit d'une impasse pour perdre un peu le joueur. Rien d'intéressant sinon des actions inutiles et n'amenant rien d'autre qu'une perte de temps préjudiciable.

La salle des gardes (25)

Rien d'intéressant non plus : il y aura des gardes du cardinal, ennemis jurés des mousquetaires... donc éventuellement un combat gratuit en perspective...

Le boudoir de la reine (19)

M. de Lapalisse dirait que c'est là qu'on y rencontrera la reine ! C'est la première étape importante à atteindre. La "première dame de

France" y confiera le véritable objet de la mission. Mais... on n'approche pas une reine si facilement !

La "Jument bleue" (38)

On l'a prévu en tant que lieu trompeur. On pourra y faire certaines actions, en fonction du cheval choisi, mais cela ne sera qu'une illusion. On y prévoira donc une issue fatale sans effet apparent.

Calais (34)

C'est le lieu d'embarquement pour l'Angleterre. Il faudra y prévoir, bien entendu, certaines conditions de paiement, d'embarquement ou simplement d'accès.

La mer

On ne s'y arrêtera pas. Nous mettrons simplement une condition de poursuite avec issue fatale à effet immédiat ou sans effet apparent.

Douvres (35)

C'est le symétrique de Calais. Pour le retour, nous retrouverons donc à peu près les mêmes types de conditions que pour l'aller.

Le bureau des douanes (27)

Il faudra y demander une autorisation. Sans cette autorisation on ne pourra pas ultérieurement continuer son chemin dans la rue de Londres.

Le restaurant (26)

Tout mousquetaire normalement constitué doit aussi manger ! Sinon il s'agira d'un piège fatal à effet retardé.

Le champ (36)

Dans un champ, en principe, il y a des fleurs... (voir la campagne).

La grange (28)

Si l'on ne peut pas vivre sans manger, on ne peut pas non plus vivre sans dormir... La grange sera là pour cela.

La rue de Londres (37)

Comme pour Calais, Douvres et la mer, c'est là que s'effectueront les tests sur certains pièges à effet retardé semés précédemment. En outre, on y mettra certaines conditions de poursuite.

La cour de Buckingham (32)

Il s'agit aussi d'un carrefour qui ressemble à la cour de Versailles. Pour être original, nous mettrons, bien entendu, des conditions différentes.

Le secrétariat du duc (30)

On y rencontrera la dame à qui offrir les fleurs si l'on veut voir le duc.

La bibliothèque du duc (20)

Est-il besoin de préciser que l'on y rencontrera le duc si l'on a rempli les conditions au secrétariat ? Comme pour la reine, nous mettrons des conditions non fatales mais obligatoires d'approche.

LA NUMÉROTATION DES LIEUX

Pour son utilisation dans le programme, chaque lieu est repéré non pas par son nom, mais par un chiffre. En effet, lors des tests (IF... THEN...) ou des branchements conditionnels (ON... GOTO...), dont nous nous servirons obligatoirement, il est plus facile d'utiliser des chiffres que des chaînes de caractères.

L'examen du plan fait apparaître une numérotation qui peut sembler farfelue. En fait, comme nous l'avions suggéré dans le Chapitre 1, nous allons utiliser une astuce nous permettant de réduire grandement le nombre d'instructions et donc de gagner en rapidité d'exécution et en place mémoire.

Pour cela, jetons tout d'abord un bref coup d'œil sur la manière générale de programmer les changements de lieu. Pour différentes raisons (entre autres pour faire apparaître la bonne description et la bonne image), le programme a besoin de reconnaître l'endroit dans lequel le joueur se trouve. Cela sera réalisé grâce au numéro de lieu stocké dans une variable. Le changement de lieu consistera donc, entre autres, à remplacer, dans cette variable, le numéro du lieu actuel par le numéro du lieu de destination.

Remarquons tout d'abord que sur les 38 lieux il en est 16 (devant les écuries, devant la maison de la veuve, etc.) à partir desquels on peut entrer quelque part, donc 16 autres (intérieur de l'écurie, etc.) dont on peut sortir pour rejoindre les lieux précédents. Il ne reste donc que 6 lieux dans lesquels on ne rentrera ou d'où l'on ne sortira pas. On peut donc facilement imaginer que la variable lieu sera très fréquemment employée (entrer, sortir, tests, etc.).

Dans un premier temps, du point de vue de la numérotation, nous allons regrouper tous les "pas de porte" ensemble, de 1 à 16. Puis, à la suite, nous regrouperons tous les intérieurs, des numéros 17 à 32. Si l'extérieur de la chambre de d'Artagnan porte le numéro 1, l'intérieur portera le numéro 17. Si nous appelons 2 le pas de porte de la maison de la veuve, l'intérieur de cette maison portera le numéro 18, etc. Nous remarquons donc qu'entre l'intérieur d'un lieu et l'extérieur correspondant il existe une différence de 16 (1 + 16 = 17, 2 + 16 = 18, etc.). Avec ce système de numérotation, si nous appelons C la variable enregistrant les lieux, deux instructions seulement seront nécessaires pour répondre à toutes les possibilités d'entrée-sortie : C = C + 16 pour le verbe *entrer* et C = C – 16 pour le verbe *sortir*. Après certains tests éventuels, quel que soit l'endroit et selon le cas, le programme se branchera toujours sur l'une ou l'autre ligne. Ce système amène une économie non négligeable d'instructions.

Cette manière de faire offre un autre avantage : quel que soit le lieu où il se trouve, chaque fois que le joueur manifestera le désir d'entrer quelque part, nous effectuerons un test pour savoir s'il n'est pas déjà à l'intérieur (et inversement pour le verbe sortir). Avec ce type de numérotation, ces tests ne nécessiteront que deux lignes simples, du type :

```
IF C > 16 THEN ...
```

(On ne peut entrer, puisqu'on est déjà à l'intérieur.)

```
IF C < 17 THEN ...
```

(On ne peut sortir, puisqu'on est déjà à l'extérieur.)

Remarquons qu'il est inutile de borner le second test en ajoutant AND C > 32, puisque dans les lieux 33 à 38 l'utilisation du verbe sortir est aussi interdite.

Afin de simplifier encore, nous allons pousser plus loin la répartition de la numérotation des lieux. On peut déjà imaginer que les pas de porte se diviseront en deux : ceux à partir desquels on pourra entrer

librement, et ceux qui exigent une condition ou qui entraînent une conséquence. Pour les mêmes raisons de simplification des tests que précédemment, les lieux à entrée conditionnelle seront numérotés de 1 à 4, tandis que les lieux à entrée libre porteront les numéros 5 à 16.

Finalement, nous pouvons résumer cette numérotation particulière par la Figure 3.

	1 Devant chambre d'Artagnan	
	2 Devant maison veuve	
ENTRÉE	3 Devant boudoir reine	
CONDITIONNELLE	4 Devant bibliothèque duc	
	5 Devant écuries	
	6 Devant usurier	
	7 Devant estaminet	SORTIE
	8 Devant cuisine	INTERDITE
	9 Devant salle gardes	(EXTERIEUR)
	10 Devant restaurant	
ENTRÉE LIBRE	11 Devant douanes	
	12 Devant grange	
	13 Devant librairie	
	14 Devant secrétariat	
	15 Devant cour Versailles	
	16 Devant cour Buckingham	
	17 Dans chambre d'Artagnan	S. CONDITIONNELLE
	18 Dans maison veuve	
	19 Dans boudoir	
	20 Dans bibliothèque	SORTIE LIBRE
	21 Dans écuries	
	22 Dans échoppe usurier	
ENTRÉE INTERDITE	23 Dans estaminet	S. CONDITIONNELLE
(INTÉRIEUR	24 Dans cuisines	
DES PIECES)	25 Dans salle gardes	
	26 Dans restaurant	
	27 Dans douanes	
	28 Dans grange	SORTIE LIBRE
	29 Dans librairie	
	30 Dans secrétariat	
	31 Dans cour Versailles	
	32 Dans cour Buckingham	
	33 Campagne	
	34 Calais	
ENTRÉE INTERDITE	35 Douvres	SORTIE INTERDITE
(PAS DE PIECES)	36 Champ	(PAS DE PIECES)
	37 Rue de Londres	
	38 Jument bleue	

Figure 3 : Liste des lieux.

3. LES CONDITIONS

Nous présentons ci-dessous l'intégralité des lieux, dans l'ordre où l'on peut les rencontrer dans le déroulement normal d'une partie (et non pas par numéro chronologique). Pour chacun de ces lieux, nous donnerons la liste des objets éventuellement présents, l'endroit où l'on peut les trouver, ce qu'il faut faire pour les obtenir, et enfin nous ferons la description complète des pièges présents et futurs.

Ces objets pourront être pris dans un ordre quelconque. Bien entendu, on pourra aussi décider de les laisser où ils sont. Dans ce cas, on pourra continuer le parcours et revenir les chercher ultérieurement (au détriment du temps passé). De toute façon, si on ne les possède pas dans certains lieux précis ou en présence de situations particulières, on se trouvera pénalisé, voire jeté en prison ou tué. Si l'on possède déjà un objet, on ne peut le prendre une deuxième fois, même si on l'a donné.

D'une manière générale, en France on parle français et on paie en bon argent français (pistoles), tandis qu'en Angleterre on parle anglais et on paie avec des livres ! On ne peut acquérir quelque chose que si l'on a de l'argent du pays et en quantité suffisante.

Outre les pièges classiques, et comme le but final doit être atteint dans un temps limité, on prévoira un "compteur de temps passé". Cela sera réalisé en incrémentant de 1 une mémoire (C0 %) chaque fois que le joueur proposera une action. Des tests seront effectués en certains endroits du programme pour savoir si cette variable n'est pas supérieure à une certaine valeur. Si c'est le cas, on aura perdu et le jeu sera arrêté.

On ne peut absolument pas entrer dans un lieu à cheval, sauf dans les cours de Versailles et de Buckingham (en venant de l'extérieur). Comme corollaire, on ne peut pas monter à cheval à l'intérieur d'une pièce. Ces tests s'effectueront sur les verbes *entrer* et *monter*. Il ne sera possible d'entrer quelque part que si l'on se trouve sur le seuil.

Nous n'avons pas prévu d'insultes. Si le lecteur le désire, il lui sera toujours possible d'ajouter, dans les dictionnaires des verbes et des noms, les mots qu'il souhaite. Afin de simplifier les tests, ces mots pourront être ajoutés tout à fait à la fin de ces dictionnaires. Il suffira alors simplement d'insérer, à la fin des deux boucles de recherche d'occurrences, les lignes suivantes :

```
198 IF VB > 34 AND C33 % = 1 THEN... (traitement des
insultes françaises)

ELSE IF VB > 34 THEN... (traitement des injures anglaises)
```

285 IF NM > 35 AND C33 % = 1 THEN... (traitement des insultes françaises)

ELSE IF NM > 18 THEN... (traitement des injures anglaises)

Attention toutefois à la place mémoire ! Comme nous le verrons dans la deuxième partie de cet ouvrage, lors de la réalisation des images, nous allons largement jouer avec l'instruction MEMORY (donc sur le HIMEM) ! L'implantation de nouvelles conditions, qui se traduit par l'ajout de nouvelles instructions dans le programme, risque donc de vous obliger à supprimer autre chose.

A l'intérieur d'une pièce, on ne pourra pas indiquer de direction (nord, sud, etc.). Ces ordres de déplacement sont réservés pour l'extérieur.

Les déplacements à pied entre les différents lieux ne seront permis que s'ils se font avant la campagne (33). En effet, la chambre de d'Artagnan, la rue de Trousse-Chemise, les écuries, l'usurier, l'estaminet et la maison de la veuve se trouvent dans Paris, donc proches les uns des autres. Par contre, partout ailleurs, pour se déplacer d'un endroit à un autre il sera nécessaire d'être à cheval.

Tous les combats, sans exception, se passeront de la manière suivante :

☐ On ne pourra attaquer que les gardes du cardinal ou les créanciers. Dans tous les autres cas, un simple message d'appel au calme sera affiché, sans aucune autre conséquence.

☐ Lorsqu'on attaque les créanciers (chambre de d'Artagnan), et si l'on a une épée, on se fait uniquement repousser. Par contre, si l'on n'a pas cette arme, les créanciers nous font jeter en prison et le jeu se termine brutalement.

☐ Lorsque le combat a lieu contre les gardes du cardinal (estaminet, salle des gardes), il existe plusieurs possibilités. Si l'on ne possède pas d'épée et que l'on n'est pas accompagné des mousquetaires, on sera arrêté et jeté en prison (fin du jeu). Si l'on a soit l'épée, soit les mousquetaires (mais pas les deux), on sera blessé et on perdra du temps à se faire soigner (le programme rajoutera 15 au nombre d'actions). Dans le cas où les mousquetaires nous accompagnent et que l'on a une épée, on se fera simplement repousser, et rien de fâcheux n'arrivera.

Chambre de d'Artagnan (17)

On peut y voir une cheminée, une patère, une armoire, une porte, un coffre et un bureau. On y trouve la majorité des objets dont on aura besoin ultérieurement au cours du jeu. Lorsqu'on aura ouvert l'armoire, le coffre ou le tiroir et que l'on aura pris les objets qui s'y trouvent, on ne pourra plus les rouvrir.

Une épée

Elle est pendue à la patère et directement accessible sans condition préalable. Elle servira dans le cas où l'on décide d'attaquer les gardes du cardinal ou les créanciers.

Une convocation de la reine

On la trouve sur le bureau, à côté du sauf-conduit. Elle n'est accessible que si l'on regarde le bureau avant de la prendre. Dans le cas contraire, si l'on n'a pas vu le bureau, l'ordinateur répondra invariablement qu'il ne sait pas où elle se trouve.

Sans que cela soit une nécessité pour la poursuite du jeu, on pourra la lire pour connaître le premier but de sa mission : aller à Versailles rencontrer la reine. Il faudra obligatoirement la présenter à un garde, lors du trajet aller, si l'on veut passer de la cour de Versailles au boudoir de la reine.

Un sauf-conduit

Il est lui aussi posé sur le bureau. Exactement de la même manière que pour la convocation, il n'est accessible que si l'on regarde préalablement le bureau.

Il ne servira que lors du trajet retour. Il devra obligatoirement être présenté au garde pour pouvoir rejoindre la porte du boudoir de la reine à partir de la cour de Versailles.

Un chapeau

Il se trouve dans le coffre. Selon le même processus que pour le bureau, on ne pourra l'obtenir que si l'on a, au préalable, ouvert ce coffre.

A priori, puisqu'il s'agit d'une pièce vestimentaire plus ou moins ornementale, on peut penser qu'il s'agit d'un leurre, donc qu'il est inutile de l'avoir sur soi. Il n'en est rien, puisque si l'on ne le possède pas lorsque l'on se trouve sur le bateau, pendant le trajet aller entre la France et l'Angleterre, on mourra d'insolation !

Une cape

Elle est cachée dans l'armoire. Comme pour le coffre, on ne pourra la prendre que si l'armoire est ouverte.

Là non plus, il ne s'agit pas d'un leurre. Elle sera nécessaire, pour se protéger du froid, lorsqu'il faudra dormir dans la grange. Dans le cas où l'on ne la possède pas, on mettra trois jours pour soigner un rhume carabiné, donc il sera impossible de mener à bien la mission (fin du jeu).

Des bijoux

On ne pourra les prendre dans le tiroir du bureau que si celui-ci est ouvert.

Il s'agit d'un piège à double détente : il faudra penser à les échanger contre des pistoles lorsqu'on se trouvera en présence de l'usurier. Cet argent servira ensuite à acquérir diverses choses utiles, nuisibles ou simplement obligatoires, ou encore à payer certains services.

Si on n'ouvre pas la porte, on ne peut sortir ! Cela semble tout à fait évident, encore faut-il penser à inclure ce test dans le programme ! Bien entendu un piège attend d'Artagnan s'il ouvre cette porte : obstinément, ses créanciers font le siège de sa chambre et n'en partiront jamais. Il sera offert la possibilité de donner certains objets (bijoux, argent, etc., si on les a) sans que cela serve à quelque chose. Les objets donnés ne pourront plus être récupérés. Malgré cela, on pourra continuer sa progression tout à fait normalement, jusqu'au moment où le jeu se terminera obligatoirement, faute d'un objet précédemment donné. Contrairement à d'Artagnan, les créanciers ne pourront prendre aucune initiative, donc, entre autres, ils n'ont pas la possibilité d'attaquer. Par contre, comme il est indiqué au début de ce chapitre, ils réagiront si d'Artagnan les attaque. La seule issue possible pour entrer ou sortir de la chambre reste la cheminée...

Rue de Trousse-Chemise (1)

Une fois sorti de la chambre de d'Artagnan, on se trouve dans cette rue. Il n'y a rien de spécial, sinon que l'on ne peut entrer dans ladite chambre que par la cheminée et à pied. Elle ne sert que de carrefour. Les seules directions autorisées et libres sont le nord (vers les écuries), le sud (vers l'échoppe de l'usurier) et l'est (vers l'estaminet).

Devant l'écurie (5)

L'entrée dans l'écurie est libre si l'on est à pied. La seule direction possible est vers le sud (vers la rue de Trousse-Chemise). Elle n'est assujettie à aucune condition.

Dans l'écurie (21)

Lorsque l'écurie est pleine, on peut y trouver quatre chevaux que nous appellerons Bonami (n° 1), Flambeau (n° 2), Laflèche (n° 3) et Mirandole (n° 4). Si l'on ne possède pas déjà un cheval, on peut prendre n'importe lequel, librement.

Le joueur pourra revenir à l'écurie pour prendre ou changer son cheval à n'importe quel moment et à partir de n'importe quel lieu. Pour cela, il faudra d'abord éventuellement laisser l'ancien pour pouvoir prendre le nouveau.

Ce que le joueur ne sait pas, c'est que Bonami est malade, que Flambeau est trop lent pour permettre d'atteindre le but à temps et que Mirandole est mal ferré. Le seul qui soit en assez bonne santé pour terminer le parcours est donc Laflèche. Bien entendu, le test sur le numéro du cheval ne se fera que bien plus loin. Nous l'avons prévu à Calais, sur le chemin aller. Si l'on ne possède pas le bon cheval, le jeu se terminera brutalement à cet endroit. Pour être tout à fait sadique, on va jusqu'à ajouter un lieu (la Jument bleue, voir ce lieu) dans lequel on pourra éventuellement réparer l'erreur de choix, mais cela sera un leurre.

La sortie de l'écurie est libre.

Devant l'échoppe de l'usurier (6)

L'entrée à pied est libre. La seule direction possible est vers le nord et elle n'est soumise à aucune condition.

Dans l'échoppe de l'usurier (22)

L'entrée à pied ainsi que la sortie ne font l'objet d'aucune condition.

Si l'on a pensé à prendre les bijoux, c'est ici qu'il faudra les échanger pour obtenir quelque argent français. Dans un deuxième temps, il sera nécessaire de changer une partie des pistoles (si l'on en possède ou s'il en reste suffisamment), afin d'avoir des livres anglaises. On ne pourra obtenir de l'argent en aucun autre lieu de quelque manière que ce soit. Bien entendu, si on ne l'a pas fait précédemment, il sera toujours possible de revenir d'un endroit quel-

conque pour changer soit les bijoux, soit les pistoles. Enfin, on ne pourra changer qu'une seule fois les bijoux ou les pistoles.

Il sera obligatoire d'acheter certains objets ou de payer pour certains services si l'on veut continuer le parcours. D'un autre côté, il existera des leurres qui ne serviront qu'à faire dépenser de l'argent pour rien. Et comme on ne possède au départ que le strict minimum... on tournera en rond sans pouvoir conclure !

Voici les endroits (et le montant des dépenses associées) dans lesquels il sera nécessaire ou facultatif de payer :

DÉPENSES OBLIGATOIRES	
Estaminet	50 pistoles
Bateau	300 pistoles
Restaurant	10 livres
Librairie	2 livres
Bateau	30 livres

Il faudra donc nécessairement avoir 350 pistoles et 42 livres anglaises pour pouvoir arriver à la fin du jeu.

DÉPENSES FACULTATIVES	
Jument bleue	200 pistoles
Rue de Londres	5 livres

Si l'on fait le compte total, cela représente donc une somme de 550 pistoles et 47 livres anglaises. Supposons que la livre vaille 10 pistoles. Le montant total des dépenses que l'on peut effectuer est donc de 550 + 470 = 1 020 pistoles. D'autre part, en ne comptant que les dépenses nécessaires, il faudra posséder l'équivalent de 350 + 420 = 770 pistoles. Nous n'en accorderons donc que 900 lors du troc des bijoux. De la même manière, nous ne donnerons que 45 livres (l'équivalent de 450 pistoles) sur les 47 possibles. Si nous résumons, l'échange des bijoux nous rapportera 900 pistoles, puis, lorsque nous aurons changé l'argent français, nous nous retrouverons à la tête d'un capital de 450 pistoles et 45 livres.

Si, sur le trajet aller, une action à la Jument bleue est décidée, quoi qu'on fasse on ne pourra prendre le bateau, puisqu'il ne nous restera que 250 pistoles (200 si l'on a payé les consommations dans l'estaminet). Symétriquement, en Angleterre, si l'on décide d'acheter les fleurs on ne pourra pas non plus reprendre le bateau puisqu'il nous manquera 2 livres. Rappelons, à cet effet, que les dépenses dans le restaurant et dans la librairie sont obligatoires. Si on ne les a pas effectuées, on

ne pourra pas, comme nous le verrons plus loin, atteindre Douvres, donc a fortiori, le bateau.

Devant l'estaminet (7)

On peut y entrer librement à pied. Comme nous allons le voir, la sortie est éventuellement conditionnelle. On ne peut aller qu'à l'ouest et à l'est (pas de conditions).

Dans l'estaminet (23)

On y trouvera d'un côté des gardes du cardinal, de l'autre Athos, Porthos et Aramis et enfin un patron.

Le but, pour d'Artagnan, est de rencontrer ses amis. Une fois entré, deux directions sont possibles : *aller à gauche* ou *aller à droite.*

Dans le premier cas, il trouvera les gardes.

Dans le second cas, il verra les mousquetaires qui décideront, à ce moment-là, de l'accompagner. A partir de cet instant, il ne pourra sortir que s'il paie les consommations de ses amis. Pour cela, il devra, au préalable, appeler le patron.

S'il décide de faire le trajet sans être accompagné par les mousquetaires, il se fera dépouiller ultérieurement de ses biens. Il ne pourra donc terminer sa mission.

Devant la maison de la veuve (2)

Les seules directions possibles sont vers l'ouest (l'estaminet) et vers l'est (campagne). Seul le déplacement vers l'ouest est libre. On ne peut se déplacer vers l'est que si l'on est à cheval. A partir de cet endroit, on ne pourra plus se déplacer à pied pour passer d'un lieu à un autre.

Dans la maison de la veuve (18)

Y entrer (à pied, bien entendu) n'amènera strictement rien pour la poursuite du jeu. La seule action possible et facultative est d'embrasser la veuve. Bien sûr, il s'agit là d'un piège fatal à effet immédiat.

La campagne (33)

Il s'agit encore d'un carrefour d'où l'on peut aller vers le nord (cour de Versailles), vers le sud (la Jument bleue) ou vers l'est (Calais).

Pour voir le duc, il faudra offrir un bouquet à sa secrétaire. On

trouvera ici des fleurs que l'on pourra cueillir. Cependant, il s'agit d'un leurre : vu le trajet restant à effectuer, elles seront fanées à l'arrivée.

On pourra aussi lire un panneau qui, dans ce cas, indiquera l'existence et la direction de la Jument bleue.

Devant la Jument bleue (38)

L'entrée dans la Jument bleue est interdite. Tout se passera donc sur le seuil et toutes les actions seront des leurres. Il sera possible d'y soigner le cheval malade, de ferrer le cheval mal chaussé ou de changer n'importe quel cheval. Aucun message d'avertissement préalable n'étant prévu, on ne s'apercevra de l'intérêt de cette station qu'après avoir perdu. De toute façon, chaque action coûte 200 pistoles et on n'aura plus assez d'argent pour prendre le bateau.

Il n'existe qu'une direction : vers le nord (à cheval).

Devant la cour de Versailles (15)

Il n'y a rien à en dire, si ce n'est que la seule direction possible se trouve au sud (à cheval). Il faut donc utiliser le verbe *entrer* si l'on veut aller dans la cour. Cette entrée pourra se faire à cheval.

Dans la cour de Versailles (31)

Il s'agit encore d'un carrefour donnant accès à la salle des gardes (nord), aux cuisines (ouest) et au boudoir de la reine (est). Ces trois derniers accès ne peuvent se faire qu'à pied. Par contre, on peut monter à cheval lorsqu'on est dans la cour et, si l'on désire aller à la campagne, il faudra utiliser le verbe *sortir* et non pas indiquer la direction sud.

Seul l'accès de la cour vers le boudoir de la reine sera conditionnel : un garde en interdira l'accès. Pour y accéder, il faudra montrer, pendant le trajet aller, la convocation et, sur le trajet retour, montrer le sauf-conduit. Bien sûr, l'inverse n'autorisera pas le passage. Par contre, l'accès du boudoir de la reine vers la cour est libre.

Devant les cuisines (8), devant la salle des gardes (9)

Rien de particulier : l'accès à pied vers ces lieux ainsi que l'entrée dans les pièces ne comportent aucune condition.

Dans les cuisines (24)

On ne pourra rien y faire et il ne se passera rien ! L'intérêt d'un tel lieu est uniquement d'accroître le sentiment d'insécurité du joueur et de lui proposer un casse-tête (qu'est-ce qu'il va m'arriver si je ne trouve pas ce qu'il faut faire ou prendre ?).

Dans la salle des gardes (25)

Les gardes du cardinal y attendent les mousquetaires. La seule action qui sera autorisée est l'attaque éventuelle avec les conséquences prévues plus haut.

Devant le boudoir de la reine (3)

On ne pourra y entrer que si l'on frappe à la porte et que l'on utilise ensuite le verbe *entrer*. Dans un deuxième temps, si la reine nous a déjà donné une bague, on ne pourra plus entrer.

Dans le boudoir (19)

Devant un personnage important, il faut être poli. Rien ne pourra donc se passer et aucune action ne pourra être entreprise si l'on n'a pas, au préalable, fait la révérence. A partir de là, la reine donnera, d'une part, l'objet final de la mission, et, d'autre part, une bague qu'il faudra montrer au duc de Buckingham pour se faire connaître.

Avec ou sans bague, la sortie du boudoir ainsi que le retour vers la cour sera libre.

Calais (34)

Dans le sens Calais –> campagne, le passage est libre. Par contre, le passage campagne –> Calais est assujetti à trois conditions :

1. être à cheval ;
2. avoir un cheval valable : soit Laflèche, soit, en fonction du mauvais cheval, avoir effectué une action à la Jument bleue ;
3. n'avoir pas trop perdu de temps. Cela se traduit, pour le programme, par le fait d'avoir compté moins de 90 actions.

Une fois ces exigences remplies, on arrive sur le port de Calais. A part revenir sur ses pas, la seule issue est vers la mer. Pour cela, il faudra obligatoirement appeler d'abord le capitaine du bateau, puis payer le prix. Il est évident qu'on ne pourra payer que si l'on possède

suffisamment d'argent. Dans le cas contraire, bien qu'il ne puisse pas embarquer, le joueur pourra continuer à se promener entre Calais et la chambre de d'Artagnan, jusqu'à son épuisement ou parce qu'il aura atteint un nombre d'actions trop important s'il revient à Calais.

En mer

C'est un lieu fictif vis-à-vis du joueur, en ce sens qu'il s'agit d'une transition permettant de changer de dictionnaire et d'effectuer la série de tests suivants :

1. Que l'on soit à Calais ou à Douvres, on ne peut traverser sans avoir appelé le capitaine et sans avoir payé le prix demandé.
2. Si les trois mousquetaires n'accompagnent pas d'Artagnan, l'équipage volera l'argent et la bague. On atteindra cependant Douvres et l'on pourra continuer à jouer sans pour cela arriver à la conclusion.
3. Si l'on n'a pas pris son chapeau, on mourra d'insolation.

Dans le cas où les tests sont passés avec succès, on se retrouve à Douvres. Bien que les mêmes tests existent aussi pour le retour Douvres –> Calais, ils seront dans ce cas inopérants, puisqu'on les aura réussis à l'aller.

Douvres (35)

Du point de vue du programme, c'est le symétrique de Calais. En venant de France, la traversée de Douvres est libre. Par contre, en sens inverse (en venant du champ), on y trouvera certains tests :

1. Si l'on n'a pas changé de cheval dans la cour de Buckingham, l'animal meurt de fatigue en arrivant à Douvres et on a perdu.
2. Si l'on n'a pas dormi dans la grange, c'est nous qui nous écroulons de fatigue.
3. Si l'on n'a pas mangé dans le restaurant, on met trois jours pour se remettre de crampes d'estomac.
4. Si le nombre d'actions effectuées est supérieur à 200, on aura perdu trop de temps.

Une fois cette barrière passée, on retrouve les mêmes séries de conditions d'embarquement qu'à Calais (appeler le capitaine, payer si l'on a assez d'argent).

Devant les douanes (11)

Rien de particulier, si ce n'est qu'on ne peut rentrer dans la pièce à cheval.

Dans le bureau des douanes (27)

Il faudra simplement penser à demander son *residence permit* pour pouvoir ultérieurement traverser la rue de Londres et aller vers Buckingham (piège à effet retardé). C'est la seule action qui sera possible ici.

Devant le restaurant (10)

Voir Devant le bureau des douanes (11).

Dans le restaurant (26)

Il sera nécessaire d'y manger (en payant) pour réussir le test lors du retour à Douvres.

Le champ (36)

Il dessert librement (à condition d'être à cheval) la grange et la rue de Londres. Nous avons vu plus haut les conditions à remplir pour aller à Douvres.

Dans ce champ, on trouvera encore des fleurs. C'est celles-ci qu'il faudra cueillir pour les offrir à la secrétaire du duc. Toutes les autres sont des leurres.

Devant la grange (12)

Voir Devant le bureau des douanes (11).

Dans la grange (28)

Si l'on est à pied, l'entrée et la sortie de la grange sont des actions tout à fait inconditionnelles. Le seul intérêt de ce lieu est de comporter un piège à effet retardé : si l'on n'y a pas dormi, il y aura une issue fatale lors du retour à Douvres. D'autre part, cela sera aussi l'aboutissement du piège fatal à effet retardé que nous avons semé dans la chambre de d'Artagnan : si l'on a oublié de prendre la cape, on met trois jours pour soigner le rhume attrapé pendant que l'on dort.

Rue de Londres (37)

L'accès à cheval est libre en venant du champ ou de la librairie. De la même manière, le retour vers ces lieux ne comporte pas de condition.

Dans cette rue on trouvera une marchande de quatre-saisons vendant des fleurs. Celles-ci pourront être acquises en payant et offertes à la secrétaire qui les acceptera. Par contre, on n'aura plus assez d'argent pour le retour.

L'accès vers le palais de Buckingham (outre le fait d'être à cheval) est soumis à deux conditions : avoir un permis de séjour délivré par le douanier, sinon un policier nous arrêtera définitivement, et posséder un plan de Londres, que l'on peut acheter à la librairie, sinon on se perd dans les rues de Londres pendant trois jours.

Devant la librairie (13)

Voir Devant le bureau des douanes (11).

Dans la librairie (29)

Après y être entré, il faudra demander un plan de Londres. Toute autre demande ou action n'aura aucun effet.

Devant la cour de Buckingham (16)

Les conditions de passage extérieur –> intérieur sont exactement les mêmes que pour la cour de Versailles.

Dans la cour de Buckingham (32)

Elle permet l'accès à pied, sans condition, vers le secrétariat au nord et vers la bibliothèque du duc à l'est. D'autre part, il faudra obligatoirement y changer de cheval si l'on veut pouvoir revenir à Douvres.

Devant le secrétariat du duc (14)

On y est déjà à pied, donc l'entrée dans le secrétariat est totalement libre.

Dans le secrétariat (30)

Pour voir le duc, il faut d'abord rencontrer la secrétaire pour lui offrir des fleurs. Seules celles qui ont été cueillies dans le champ ou

achetées dans la rue de Londres seront acceptées. Celles qui viennent de la campagne seront refusées.

Devant la bibliothèque (4)

Pour entrer dans cette pièce, il faut obligatoirement frapper à la porte, avoir offert les bonnes fleurs à la secrétaire et utiliser le verbe *entrer*.

Dans la bibliothèque (20)

Le duc n'acceptera de remettre les ferrets que si l'on a, dans l'ordre, fait la révérence, montré la bague et demandé les ferrets.

Il ne restera plus alors qu'à faire le trajet en sens inverse sans oublier de changer de cheval dans la cour de Buckingham.

Sur le trajet de retour, comme nous l'avons vu, les seuls écueils que l'on rencontrera seront situés à l'entrée de Douvres (test sur le cheval utilisé, sur le fait d'avoir dormi, mangé, et sur le nombre d'actions effectuées).

Arrivé dans la cour de Versailles, il sera nécessaire de descendre de cheval, de montrer le sauf-conduit (et non plus la convocation), d'aller vers l'est, de frapper à la porte et d'entrer.

Afin d'être traitées informatiquement, toutes les conditions seront stockées dans des variables sous la forme de condition logique (0 = faux, 1 = vrai). Dans certains cas, cependant, il sera nécessaire de dépasser le chiffre 1 : le joueur a la possibilité de choisir un cheval parmi quatre. Nous enregistrerons donc le numéro du cheval choisi dans une seule variable (C11 %). Lorsqu'on aura fait une action à la Jument bleue, cette variable sera mise à 5. Si l'on a changé sa monture dans la cour de Buckingham, C11 % prendra la valeur 6. De la même manière, si l'on ne possède pas les bijoux, C5 % sera à 0 ; lorsqu'on aura pris les bijoux, elle sera mise à 1 et, dans le cas où on les aura donnés aux créanciers, par exemple, elle comportera la valeur 3. Cette manière de faire permet d'enregistrer, pour une condition particulière, toutes les situations ou tous les cas possibles, donc d'effectuer tous les tests nécessaires.

Enfin, on peut remarquer qu'on ne mettra, dans ces variables, que des valeurs entières. Or, dans la majorité des cas, les ordinateurs (dont l'Amstrad) ont besoin de moins d'octets pour stocker les variables

entières que pour les variables numériques. C'est pour cette raison que toutes les variables enregistrant les conditions sont déclarées entières.

La liste complète des variables de conditions avec les valeurs qu'elles peuvent prendre est présentée en Figure 4. Une première liste avait été constituée en même temps que l'élaboration des conditions détaillées qui précèdent et avant la programmation réelle. Au fur et à mesure de la réalisation du programme BASIC, elle a été légèrement modifiée, afin de satisfaire les besoins non prévus ou au contraire excédentaires, et en arriver ainsi progressivement à la liste définitive.

C	(0 à 38)	Numéro du lieu
C0 %	(0 à X)	Nombre d'actions
C1 %	(0 à 1)	Vu bureau
C2 %	(0 à 1)	Pris convocation
C3 %	(0 à 1)	Pris sauf-conduit
C4 %	(0 à 1)	Ouvert tiroir
C5 %	(0 à 3)	Pris / donné / changé bijoux
C6 %	(0 à 1)	Ouvert armoire
C7 %	(0 à 1)	Pris cape
C8 %	(0 à 1)	Ouvert coffre
C9 %	(0 à 1)	Pris chapeau
C10 %	(0 à 2)	Pris / donné épée
C11 %	(0 à 6)	Numéro cheval choisi / soigné / changé
C12 %	(0 à 1)	Obtenu ferrets
C13 %	(0 à 900)	Nombre de pistoles
C14 %	(0 à 1)	Changé pistoles
C15 %	(0 à 45)	Nombre de livres
C16 %	(0 à 1)	Vu mousquetaires
C17 %	(0 à 1)	Appelé patron
C18 %	(0 à 1)	Payé consommations
C19 %	(0 à 1)	Reste cheval 1 à l'écurie
C20 %	(0 à 1)	Reste cheval 2 à l'écurie
C21 %	(0 à 1)	Reste cheval 3 à l'écurie
C22 %	(0 à 1)	Reste cheval 4 à l'écurie
C23 %	(0 à 1)	Montré bague
C24 %	(0 à 1)	Acheté plan
C25 %	Non utilisé	
C26 %	(0 à 1)	Monté cheval
C27 %	(0 à 1)	Avoir dormi
C28 %	(0 à 1)	Avoir mangé
C29 %	(0 à 1)	Acheté carte de séjour
C30 %	(0 à 1)	Frappé porte
C31 %	(0 à 1)	Fait révérence
C32 %	(0 à 1)	Avoir bague
C33 %	(0 à 1)	Etre en France / Angleterre
C34 %	(0 à 1)	Ouvert porte
C35 %	(0 à 3)	Avoir fleurs
C36 %	(0 à 1)	Cueilli fleurs
C37 %	(0 à 1)	Montré convocation / sauf-conduit
C38 %	(0 à 1)	Donné fleurs

Figure 4 : Liste des conditions.

4. TRAITEMENT DES ACTIONS

Une fois l'analyse de syntaxe effectuée, il existe deux possibilités de traitement des actions : soit, pour un *lieu* donné, traiter toutes les actions possibles ou permises, soit, pour une *action* donnée, traiter tous les lieux possibles.

Dans le premier cas, le branchement s'effectuera en fonction du numéro du lieu vers un sous-programme correspondant dans lequel on examinera tous les cas de tous les verbes et de tous les noms autorisés pour ce lieu.

Dans le second cas, le branchement s'effectuera en fonction du numéro de verbe vers un sous-programme dans lequel seront traités tous les lieux pour lesquels l'action est permise.

Pour le premier type de traitement, les différents tests faits sur la plupart des verbes se répéteront de façon plus ou moins identique autant de fois qu'il y a de lieux. On peut alors imaginer, afin de gagner de la place mémoire, que pour un lieu déterminé chaque verbe fasse l'objet d'un branchement vers un sous-programme commun... et nous sommes pratiquement ramenés au second type de traitement, par suppression du passage dans les lieux !

Nous avons donc choisi la seconde solution. Cependant, pour certains endroits du jeu dans lesquels on utilise des verbes ou des traitements tout à fait spécifiques (le bateau, Calais et Douvres), il nous a paru intéressant de conserver le premier type de traitement. Pour tous les autres cas, on réalisera autant de sous-programmes qu'il y aura de verbes. C'est à l'intérieur de ces modules que se fera le traitement des actions en fonction du lieu.

Pour des raisons de taille de livre et d'intérêt pour le lecteur, il est difficilement concevable de passer tous les modules en revue. Nous vous présentons ci-après les plus représentatifs afin d'en expliquer la conception et l'architecture. Les autres modules, étant basés sur le même principe, pourront s'en déduire facilement.

Nous avons vu que le branchement vers ces différents modules s'effectuait à la fin de l'analyse de syntaxe (ligne 290).

■ MANGER

Pour chaque verbe, dans un premier temps, il convient de bien préciser en clair, et sans en oublier, les différents cas d'emploi ainsi que les hypothèses retenues :

- On ne peut pas manger dans un lieu autre que le restaurant (C = 26), que l'on soit en France ou en Angleterre.

- Si l'on n'a plus assez d'argent sur soi, on ne peut pas non plus manger.
- Si l'on n'est pas dans un des cas précédents, alors on peut manger et le montant du repas est déduit de la somme d'argent disponible.

Une fois les hypothèses et les cas ci-dessus bien vérifiés, nous pouvons les transcrire en programme :

```
15000 IF C<>26 THEN IF C33%=1 THEN PRINT
 "CE N'EST PAS UN RESTAURANT, ICI":GOTO
10 ELSE PRINT "YOU ARE NOT IN A RESTAURA
NT, HERE":GOTO 10
15010 IF C15%<10 THEN 16050
15020 C15%=C15%-10:C28%=1:PRINT "AH ! YO
U FEEL BETTER":PRINT "BUT YOUR PURSE LOS
T 10 POUNDS":GOTO 10
```

■ ALLER

De la même manière que précédemment, les hypothèses et les différents cas sont les suivants :

- A l'exception de l'estaminet (C = 23), si l'on se trouve à l'intérieur d'une pièce en France ou en Angleterre (C < 31 ou C > 16), on ne peut *aller* nulle part.
- Si l'on se trouve à l'extérieur d'une pièce en France ou en Angleterre (C < 17 ou C > 30), on ne peut *aller* que dans une direction géographique (nord, sud, est ou ouest).
- A l'intérieur de l'estaminet (seul lieu restant à traiter puisque nous avons épuisé tous les autres dans les deux conditions précédentes), on ne peut aller qu'à droite ou à gauche.
- Si l'on va à gauche, on trouve les gardes du cardinal.
- Toujours à l'intérieur de l'estaminet (sous-entendu en voulant aller à droite, puisque c'est la seule direction restante), si l'on a déjà vu les mousquetaires et que l'on n'a pas payé les consommations, on ne peut aller nulle part sans avoir payé.
- Si l'on a déjà vu les mousquetaires, il n'y a plus personne.

• Le seul cas restant est celui où l'on va à droite sans avoir vu les mousquetaires. Il est alors inutile d'effectuer les tests considérés et l'on affiche directement le message voulu.

A partir de ces hypothèses, nous pouvons donc bâtir point par point notre sous-programme :

```
9500 IF C>16 AND C<31 AND C<>23 THEN IF
C33%=1 THEN PRINT "A L'INTERIEUR D'UNE P
IECE,":PRINT "ON NE PEUT ALLER NULLE PAR
T":GOTO 10 ELSE PRINT "INSIDE A ROOM THE
RE IS NO WAY TO GO":GOTO 10
9510 IF C<17 OR C>30 THEN IF C33%=1 THEN
 PRINT "DANS QUELLE DIRECTION ?":GOTO 10
 ELSE PRINT "WHICH DIRECTION ?":GOTO 10
9520 IF NM<29 OR NM>30 THEN PRINT "DE QU
EL COTE ?":GOTO 10
9530 IF NM=30 THEN IM$="CIM23G":GOSUB 64
100:PRINT "CE SONT LES GARDES DU CARDINA
L.":PRINT "ILS SONT BIEN ARMES !":GOTO 1
0
9535 IF C16%=1 AND C18%=0 THEN PRINT "PA
S SANS PAYER LES CONSOMMATIONS":GOTO 10
9540 IF C16%=1 THEN PRINT "IL N'Y A PLUS
 PERSONNE":GOTO 10
9550 IM$="CIM23D":GOSUB 64100:PRINT "CE
SONT ATHOS, PORTOS ET ARAMIS.":PRINT "AP
RES EXPLICATIONS,":PRINT "ILS DECIDENT D
E VOUS SUIVRE":C16%=1:GOTO 10
```

■ DEMANDER

Les hypothèses sont les suivantes :
Si l'on est ailleurs que dans la bibliothèque du duc (C <> 20), dans le bureau des douanes (C <> 27) ou dans la librairie (C <> 29), on ne peut rien demander.

Dans la bibliothèque du duc (C = 20)

On ne peut demander quelque chose que si l'on a fait la révérence.
On ne peut rien demander d'autre que les ferrets.
Si l'on n'a pas montré la bague, on ne peut rien obtenir.

Si l'on ne se trouve pas dans l'un des cas précédents, alors la condition *avoir ferrets* est initialisée (C12 % = 1).

Dans la librairie (C = 29)

On ne peut demander autre chose qu'un plan de Londres.

Si l'on a déjà ce plan, on ne peut le demander une deuxième fois.

On ne peut obtenir ce plan que si l'on a suffisamment d'argent sur soi.

Alors seulement, la condition *avoir plan* (C24 %) sera mise à 1 et le prix du plan déduit de l'argent disponible.

Dans le bureau des douanes (C = 27)

On ne peut demander qu'un permis de séjour.

Si l'on en a déjà un, on ne peut en obtenir un deuxième.

La condition *avoir permis de séjour* (C29 %) n'est mise à 1 que si les tests précédents sont tous négatifs.

Traduit en langage BASIC, cela donne le sous-programme suivant :

```
9000 IF C<>20 AND C<>27 AND C<>29 THEN 4
010
9010 IF C=29 THEN 9060 ELSE IF C=27 THEN
 9100
9020 IF C31%=0 THEN PRINT "YOU ARE VERY
IMPOLITE, MUSKETEER !":GOTO 10
9030 IF NM<>18 THEN PRINT "SORRY, I CAN'
T GIVE YOU THAT":GOTO 10
9040 IF C23%=0 THEN PRINT "SORRY, MONSIE
UR, I DON'T KNOW YOU":GOTO 10
9050 PRINT "OK, HERE ARE THE FERRETS.":P
RINT "PLEASE, GIVE THE QUEEN MY BEST REG
ARDS":C12%=1:GOTO 10
9060 IF NM<>17 THEN PRINT "YOU DON'T NEE
D THAT": GOTO 10
9070 IF C24%=1 THEN 4115
9080 IF C15%<2 THEN 16050
9090 C24%=1:C15%=C15%-2:PRINT "HERE IS T
HE MAP":GOTO 10
9100 IF NM<>6 THEN PRINT "I HAVE NOT THA
T THING":GOTO 10
9110 IF C29%=1 THEN 4115
9120 C29%=1:PRINT "HERE IS YOUR RESIDENC
E-PERMIT.":PRINT "THAT'S FREE.":GOTO 10
```

5. DIALOGUE HOMME-MACHINE

■ ANALYSE DE SYNTAXE

Comme nous l'avons vu, la base de toute analyse de syntaxe est un dictionnaire de verbes et de noms stocké dans la mémoire du système.

Le principe d'une analyse de syntaxe est alors très simple : il suffit de comparer un mot proposé par le joueur avec les mots du dictionnaire. S'il y a occurrence, l'aiguillage se fait vers le sous-programme de traitement spécifique.

Afin de rester compatible avec la taille limitée de la mémoire d'un ordinateur individuel sans toutefois nuire à l'intérêt du jeu, nous nous sommes fixé les choix suivants :

- La recherche d'occurrences se fera par balayage systématique du dictionnaire.
- La comparaison se fera sur les six premières lettres de chaque mot (pour pouvoir distinguer par exemple le verbe *ferrer* du nom *ferret*).
- Dans certains cas, le système pourra comprendre une seule initiale (I pour inventaire, O pour ouest, etc.).
- On commencera par effectuer la recherche d'occurrences sur les verbes avant de le faire sur les noms.
- Lorsqu'un mot est extrait de la phrase proposée par le joueur, il sera immédiatement comparé au dictionnaire.
- Lorsqu'un verbe ou un nom sera déclaré valide par le test d'occurrence, on arrêtera la recherche sur ce mot, sans continuer la comparaison avec le reste du dictionnaire.

SAISIE DE LA RÉPONSE

```
10 C0%=C0%+1:K=0:INPUT"-->",R$:R$=UPPER$
(R$)
20 IF LEN(R$)=0 THEN 10
```

Ligne 10

Chaque action décidée par le joueur passe obligatoirement par l'analyse de syntaxe. C'est donc le lieu idéal pour incrémenter le compteur d'action C0 %. L'instruction INPUT (contrairement à GET$) permet de saisir plusieurs caractères qui seront stockés dans R$. Le fait de remplacer la virgule par un point-virgule entre INPUT et R$ a

pour effet de supprimer l'apparition du point d'interrogation qui sert d'invite pour l'instruction INPUT. Enfin, dans le but d'éviter une double série de tests, l'instruction UPPER$ transformera les minuscules éventuelles comprises dans R$ en majuscules.

Ligne 20

Pour le cas où la touche ENTER serait frappée juste après le signe d'invite (–>), R$ contiendra une chaîne vide. Il sera alors inutile de poursuivre le traitement. On ajoute donc un test qui renverra à la ligne précédente si la longueur (LEN) de R$ est nulle.

TRAITEMENT DES INITIALES

```
30 IF R$="I" THEN 8000
40 IF LEN(R$)=1 THEN 500
```

Comme elles sont peu nombreuses et afin d'éviter des tests inutiles, les initiales sont traitées juste après la saisie de la réponse. Nous en avons prévu six : I pour la demande d'inventaire et cinq autres pour les déplacements géographiques (N, S, E, O et W).

Ligne 30

On se débarrasse d'abord de la lettre I. Si R$ contient uniquement cette lettre, le branchement se fera à la ligne 8000, adresse à partir de laquelle se trouve le sous-programme d'affichage de l'inventaire.

Les autres initiales n'intéressant (sauf erreur de frappe) que les déplacements, elles seront testées individuellement dans le sous-programme correspondant placé à partir de la ligne 500. Nous les traiterons donc globalement à la ligne 40 en regardant simplement si la variable R$ ne comprend qu'un seul caractère.

EXTRACTION ET FORMATAGE D'UN MOT

Si l'on fait abstraction des apostrophes et de certains signes de ponctuation, on remarque tout d'abord que tous les mots d'une phrase sont séparés par un espace. Il suffit donc de noter la position des espaces à l'intérieur de la phrase (à l'aide de l'instruction INSTR$) puis d'extraire le mot par l'instruction MID$. Il faut toutefois savoir

que lorsqu'une chaîne de caractères est saisie par un INPUT, l'appui sur la touche RETURN n'ajoute pas d'espace à la fin de la chaîne.

```
50 IF RIGHT$(R$,1)=" " THEN 70
60 R$=R$+" "
70 VB=0:VR$=""
80 RE$="":NM=0:R=0
90 A=R+1:R=INSTR(A,R$," ")
100 RE$=MID$(R$,A,R-A)
110 IF MID$(RE$,2,1)="'" THEN RE$=MID$(R
E$,3)
115 IF K=0 THEN V$=RE$ ELSE N$=RE$
120 IF LEN(RE$)>6 THEN RE$=LEFT$(RE$,6):
 GOTO 150
130 RP=ASC(RIGHT$(RE$,1))
140 IF RP<65 OR RP>90 THEN RE$=LEFT$(RE$
,LEN(RE$)-1)
```

Ligne 50

Si le dernier caractère de la phrase contenue dans R$ est un blanc, on saute la ligne 60, cette dernière ligne étant simplement destinée à ajouter un espace à la fin de la phrase.

Ligne 90

L'instruction INSTR permet de chercher dans R$, à partir de la position n° A, le numéro de position du prochain blanc. Ce numéro est stocké dans R.

Ligne 100

A l'aide de l'instruction MID$ on extrait de R$ les R–A caractères suivants à partir de la position A. Par exemple :

Position des caractères	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Contenu de R$	M	O	N	T	E	R		C	H	E	V	A	L	

(Remarquez l'espace éventuellement ajouté en position 14 par la ligne 60.)

Au départ, A vaut 1 (A = 0 + 1). Donc, l'instruction INSTR$ va

rechercher, à partir du premier caractère, le numéro de position de l'espace suivant (7) et le stocker dans R. L'instruction MID$ va ensuite extraire de R$ les caractères contenus entre A = 1 et R – A = 6 et les stocker dans RE$. Ce qui correspond bien au mot MONTER.

Après traitement de ce mot et sans changer les variables, nous reviendrons directement à la ligne 90. A ce moment, A = R + 1 = 7 + 1 = 8. Selon le même principe que précédemment, INSTR va mettre 14 dans R et MID$ chargera RE$ avec le mot CHEVAL compris entre A = 8 et R – 1 = 13.

Nous verrons plus loin comment stopper le processus lorsque le dernier mot de la phrase a été traité.

Supposons maintenant que la phrase du joueur soit la suivante : OUVRIR L'ARMOIRE, QUI...

Le traitement que nous venons de voir va extraire les caractères "L'ARMOIRE,". Il nous faut donc nous débarrasser du L' et de la virgule. Dans la phraséologie restreinte d'un jeu d'aventure, l'apostrophe se trouvera toujours en deuxième position (donc éviter les mots du genre D'ACCORD ou commençant par QU'). Il nous suffit donc de programmer la phrase suivante : "Si le deuxième caractère est une apostrophe, alors, extraire tous les caractères à partir du troisième". C'est ce qui est réalisé à la ligne 110. Remarquez que dans le deuxième MID$ nous avons indiqué la position de départ (3) sans mentionner le nombre de caractères à extraire. MID$ va donc sortir tous les caractères à partir du 3^{e}, c'est-à-dire ARMOIRE, (y compris la virgule).

Ligne 120

Dans notre cas, le problème de la virgule sera résolu en même temps que la troncature des mots après le 6^{e} caractère : si la longueur de RE$ est supérieure à 6 caractères, alors ne conserver dans RE$ que les 6 premiers. A l'issue du traitement de la ligne 120, il ne subsistera donc plus dans RE$ que le mot ARMOIR.

Lignes 130 et 140

Par contre, si la longueur du mot est inférieure ou égale à 6 caractères, on commencera par transformer (ligne 130) le dernier caractère en code ASCII. On effectuera ensuite un test sur ce code (ligne 140) pour savoir s'il est plus petit que celui de la lettre A ou plus grand que celui de la lettre Z. Si c'est le cas, le dernier caractère n'est pas une lettre et on le supprime.

RECHERCHE D'OCCURRENCES

Comme nous avons décidé de balayer le dictionnaire, la recherche d'occurrences sera incluse dans une boucle dont la borne supérieure sera égale au nombre de mots inclus dans le dictionnaire considéré. Nous aurons donc deux boucles : une pour la recherche des verbes, l'autre pour la recherche des noms. Plusieurs hypothèses seront à considérer :

☐ On traitera les verbes avant de traiter les noms.

☐ Lorsqu'une occurrence se produit, il faut pouvoir sortir de la boucle sans continuer la recherche jusqu'à la fin du dictionnaire.

☐ Lorsqu'un mot (verbe ou nom) aura été déclaré valable, on passera directement au traitement suivant, même si tous les mots de la phrase n'ont pas été testés.

☐ Si aucun verbe n'a été trouvé, on affichera un message et on retournera directement à la saisie sans faire de recherche sur les noms.

☐ Il se peut qu'un même mot, présent dans les deux dictionnaires (verbes et noms), soit extrait deux fois. Il sera donc nécessaire de ne pas le compter dans les noms.

Ces fonctions sont réalisées par quatre modules compris entre les lignes 150 à 280 listées ci-dessous. Le premier (lignes 160 à 180) et le troisième modules (lignes 220 à 250) sont respectivement les boucles de recherche d'occurrences sur les verbes et sur les noms. Le deuxième (lignes 190 à 200) et le quatrième modules (lignes 270 et 280) permettent le traitement direct de verbes ou de noms particuliers :

```
150 IF K=1 THEN 220
160 FOR I=1 TO NV
170 IF VE$(I)=RE$ THEN VB=I:VR$=VE$(I):I
=NV:K=1
180 NEXT I:IF R=LEN(R$) AND VB=0 THEN IF
 C33%=1 THEN PRINT "JE NE COMPREND PAS":
 GOTO 10 ELSE PRINT "I DON'T UNDERSTAND"
:GOTO 10
190 IF VB=11 THEN 8000
195 IF VB=33 OR VB=34 THEN 17000
200 IF VB>28 THEN R$=LEFT$(VR$,1):GOTO 5
00
210 IF K=1 THEN 80 ELSE 90
```

```
220 FOR I=1 TO NN
230 IF NO$(I)=VR$ THEN 250
240 IF NO$(I)=RE$ THEN NM=I:I=NN
250 NEXT I
260 IF R<>LEN(R$) AND NM=0 THEN 90
270 IF NM=1 THEN 8000
280 IF NM>1 AND NM<6 THEN R$=LEFT$(RE$,1
):GOTO 500
```

Ligne 150

K est un drapeau que l'on met à 1 lorsqu'un verbe a été trouvé. Si aucun verbe n'est valide, K est à zéro. Donc, avant de rentrer dans la boucle de recherche des verbes, on effectue un test sur K : s'il est égal à 1, on saute directement à la boucle de recherche des noms.

Ligne 170

Si le verbe extrait du dictionnaire (VE$(I)) est le même que celui extrait de la phrase, alors on stocke : le numéro du verbe dans VB, le verbe lui-même dans une variable provisoire VR$, la valeur finale de la boucle dans I (sortie de la boucle), 1 dans le drapeau K.

Ligne 180

Lorsqu'on est sorti de la boucle (I = NV), si la position notée dans R correspond à la fin de la phrase ET si aucun verbe n'a été reconnu, alors on affiche un message (soit en français, soit en anglais) puis on revient directement à la saisie.

Lignes 190 à 200

Si certains verbes particuliers ont été reconnus, on stoppe le processus d'analyse de syntaxe en se déroutant vers les sous-programmes correspondants.

Ligne 210

Si le drapeau est à 1 (verbe trouvé), on revient à la ligne 80 pour initialiser la recherche des noms. Sinon (verbe non trouvé), on continue l'extraction des verbes de la phrase (ligne 90).

La boucle de recherche d'occurrences des noms se trouve entre les lignes 220 et 250.

Ligne 230

Si le nom extrait du dictionnaire (NO$(I)) est le même que le verbe trouvé précédemment (stocké dans VR$), on passe directement au nom suivant dans le dictionnaire.

Ligne 240

Si le nom extrait du dictionnaire est le même que celui extrait de la phrase (RE$), alors on stocke le numéro du verbe trouvé dans NM, puis on met dans le compteur de boucle I sa valeur finale.

Ligne 260

Si l'on n'a pas examiné tous les mots de la phrase ET si l'on n'a pas trouvé de nom, alors on retourne en 90 pour continuer la recherche des noms.

Lignes 270 à 280

De la même manière que pour les verbes, on traite directement les noms particuliers.

Il faut enfin noter la ligne 115 qui est destinée à mettre dans V$ et N$ le verbe et le nom trouvés. Ces variables serviront dans l'impression des messages communs.

BRANCHEMENTS

Nous avons vu que le verbe est déterminatif d'une action et que c'est en se référant à lui (à son numéro, dans notre cas) que seront effectués les divers branchements. Cette fonction sera donc réalisée très simplement par la ligne 290 :

```
290 ON VB GOTO 3500,4000,4500,5000,5500,
6000,6500,7000,7500,7500,8000,8500,9000,
9500,10000,10500,11000,11500,12000,12500
,13000,13500,14000,14500,15000,15500,160
00,16500
```

Nous trouverons donc en 3500 le module traitant le verbe n° 1 (voir), en 4000 le module traitant le verbe n° 2 (prendre), etc. Remarquez la répétition du branchement en 7500. Cela est dû au fait que nous traitons les verbes 9 (descendre) et 10 (monter) dans le même module situé en 7500.

■ DESCRIPTION DES LIEUX

Situé entre les lignes 1000 et 1420, le sous-programme de description des lieux est basé sur le principe le plus simple de notre jeu d'aventure.

```
1005 ON C GOTO 1010,1020,1030,1040,1050,
1060,1070,1080,1090,1100,1110,1120,1130,
1140,1150,1160,1170,1180,1190,1200,1210,
1260,1270,1280,1290,1300,1310,1320,1420,
1330,1340,1350,1360,1370,1380,1390,1400,
1410
1010 PRINT "RUE DE TROUSSE-CHEMISE,":PRI
NT "DEVANT VOTRE CHAMBRE":GOTO 10
1020 PRINT "ICI HABITE UNE CHARMANTE VEU
VE...":GOTO 10
1030 PRINT "LA PORTE DU BOUDOIR DE LA RE
INE.":GOTO 10
1040 PRINT "HERE IS THE DOOR OF THE DUKE
'S LIBRARY":GOTO 10
1050 PRINT "VOICI L'ENTREE DE VOTRE ECUR
IE...":GOTO 10
1060 PRINT "L'ECHOPE DE L'USURIER...":PR
INT "DONNEZ BEAUCOUP POUR AVOIR PEU":GOT
O 10
1070 PRINT "L'ENTREE DE L'ESTAMINET DU C
OIN,":PRINT "RENDEZ-VOUS PRIVILEGIE DES
GARDES ET DESMOUSQUETAIRES":GOTO 10
```

1080 ... etc.

En ligne 1005, le programme se branche à la ligne correspondant au numéro du lieu (ON C GOTO...). Chacune de ces lignes (ou groupe

de lignes) sera essentiellement constituée de l'impression du message considéré suivi d'un retour à la saisie de la phrase du joueur. Remarquez les espaces ou les traits d'union parfois ajoutés dans certains textes : ils sont simplement prévus afin de terminer une ligne en repoussant l'impression du mot suivant sur la ligne qui suit. Cela évite ainsi d'avoir des mots tronqués aléatoirement en fin de ligne.

■ DICTIONNAIRE

Les dictionnaires, au nombre de deux, seront réalisés sous la forme de deux tableaux (verbes et noms). Il sera donc prévu deux dictionnaires pour la partie française, et deux autres pour la partie anglaise.

La liste complète des mots que nous avons choisis est indiquée en Figure 5 pour les verbes et en Figure 6 pour les noms.

On pourra noter que certains mots sont communs à la fois au dictionnaire des verbes et à celui des noms. Il s'agit des directions (nord, sud, est, ouest) et du mot *inventaire*. En voici la raison : pour la détermination d'une action, le verbe est prépondérant. C'est-à-dire que pour que l'ordinateur comprenne la demande d'action du joueur, celui-ci doit obligatoirement inclure dans sa phrase un verbe reconnu, éventuellement suivi d'un nom. Les mots ci-dessus étant employés fréquemment tout au long du jeu, il est classique, dans tout jeu d'aventure, de permettre au joueur de n'indiquer qu'une initiale de direction (N pour *nord*), qu'un mot *(nord)* ou une phrase complète *(aller nord)*. Les initiales sont traitées directement et d'une manière très simple, juste après la saisie, sans passer par la recherche d'occurrences (voir le paragraphe traitant de l'analyse de syntaxe). Ce n'est donc pas la peine de les inclure dans le dictionnaire. Lorsque le joueur indique le mot complet, il est nécessaire que l'ordinateur comprenne ce mot comme étant un verbe. Dans ce cas, ce mot devra figurer au dictionnaire des verbes. De la même manière, lorsqu'on indique une phrase complète, le mot considéré devra être inclus dans le dictionnaire des noms.

D'autre part, il est d'usage (sans que cela soit une règle absolue) d'indiquer les verbes à l'infinitif. En effet, même si l'analyse de syntaxe le permet, le joueur n'inscrira qu'une fois ou deux, et uniquement pour sa satisfaction personnelle, une phrase du genre "Je souhaiterais monter sur ce sacré cheval afin de poursuivre ma route". C'est long et fastidieux, surtout s'il faut réaliser quelques centaines d'actions pour arriver au but. Très vite, on se contentera de dire *monter à cheval.*

N°	FRANÇAIS	ANGLAIS
1	VOIR	<TO> LOOK
2	PRENDR<E>	<TO> GET
3	SORTIR	<TO COME> OUT
4	OUVRIR	<TO> OPEN
5	FERMER	<TO> SHUT
6	ATTAQU<ER>	<TO> ATTACK
7	DONNER	<TO> GIVE
8	LAISSE<R>	<TO> LET
9	DESCEN<DRE>	<TO COME> DOWN
10	MONTER	<TO COME> UP
11	INVENT<AIRE>	INVENT<ARY>
12	ENTRER	<TO COME> IN
13	DEMAND<ER>	<TO> ASK
14	ALLER	<TO> GO
15	CHANGE<R>	<TO> CHANGE
16	SOIGNE<R>	<TO> TREAT
17	FERRER	<TO> SHOE
18	PAYER	<TO> PAY
19	APPELE<R>	<TO> CALL
20	FAIRE	<TO> DO
21	MONTRE<R>	<TO> SHOW
22	FRAPPE<R>	<TO> KNOCK
23	LIRE	<TO> READ
24	CUEILL<IR>	<TO> PICK
25	MANGER	<TO> EAT
26	DORMIR	<TO> SLEEP
27	ACHETE<R>	<TO> BUY
28	EMBRAS<SER>	<TO> KISS
29	NORD	NORTH
30	SUD	SOUTH
31	EST	EAST
32	OUEST	WEST
33	SAUVER	<TO> SAVE
34	REPREND<RE>	<TO> RESTOR<E>

Figure 5 : Dictionnaire des verbes.

N°	FRANÇAIS	ANGLAIS
1	INVENT<AIRE>	INVENT<ARY>
2	NORD	NORTH
3	SUD	SOUTH
4	EST	EAST
5	OUEST	WEST
6	CONVOC<ATION>	RESIDE<NCE PERMIT>
7	CHEVAL	HORSE
8	FLEURS	FLOWER<S>
9	ARGENT	MONEY
10	LIVRES	POUNDS
11	REVERE<NCE>	BOW
12	CAPITA<INE>	CAPTAI<N>
13	BATEAU	BOAT
14	PORTE	DOOR
15	CHAMBR<E>	DUKE
16	CHEMIN<EE>	RING
17	BUREAU	MAP
18	ARMOIR<E>	FERRET<S>
19	COFFRE	
20	TIROIR	
21	PATERE	
22	CAPE	
23	CHAPEA<U>	
24	SAUF-C<ONDUIT>	
25	EPEE	
26	BIJOUX	
27	PANNEA<U>	
28	PISTOL<ES>	
29	DROITE	
30	GAUCHE	
31	PATRON	
32	BONAMI	
33	FLAMBE<AU>	
34	LAFLEC<HE>	
35	MIRAND<OLE>	

Figure 6 : Dictionnaire des noms.

Enfin, dans le dictionnaire des noms, on remarquera que certains mots anglais ne sont pas la traduction des mots français portant le même numéro. Dans la partie du jeu qui se déroule en Angleterre, nous n'avons pas prévu, par exemple, de cheminée. Il est donc inutile, et même nuisible pour l'encombrement mémoire, de traduire ce mot qui n'a que très peu de chances d'être employé.

Ces dictionnaires seront chargés, l'un remplaçant l'autre et en fonction des besoins, au moment du passage d'un pays à l'autre à l'aide de l'instruction READ associée à DATA et RESTORE.

```
2000 NV=34:IF C33%=1 THEN NN=35:RESTORE
2020 ELSE NN=18:RESTORE 2040
2010 FOR I=1 TO NV:READ VE$(I):NEXT I:FO
R I=1 TO NN:READ NO$(I):NEXT I:GOTO 1000
2020 DATA VOIR,PRENDR,SORTIR,OUVRIR,FERM
ER,ATTAQU,DONNER,LAISSE,DESCEN,MONTER,IN
VENT,ENTRER,DEMAND,ALLER,CHANGE,SOIGNE,F
ERRER,PAYER,APPELE,FAIRE,MONTRE,FRAPPE,L
IRE,CUEILL,MANGER,DORMIR,ACHETE,EMBRAS,N
ORD,SUD,EST,OUEST,SAUVER,REPREN
2030 DATA INVENT,NORD,SUD,EST,OUEST,CONV
OC,CHEVAL,FLEURS,ARGENT,LIVRES,REVERE,CA
PITA,BATEAU,PORTE,CHAMBR,CHEMIN,BUREAU,A
RMOIR,COFFRE,TIROIR,PATERE,CAPE,CHAPEA,S
AUF-C,EPEE,BIJOUX,PANNEA,PISTOL,DROITE,G
AUCHE,PATRON,BONAMI,FLAMBE,LAFLEC,MIRAND
2040 DATA LOOK,GET,OUT,OPEN,SHUT,ATTACK,
GIVE,LET,DOWN,UP,INVENT,IN,ASK,GO,CHANGE
,TREAT,SHOE,PAY,CALL,DO,SHOW,KNOCK,READ,
PICK,EAT,SLEEP,BUY,KISS,NORTH,SOUTH,EAST
,WEST,SAVE,RESTOR
2050 DATA INVENT,NORTH,SOUTH,EAST,WEST,R
ESIDE,HORSE,FLOWER,MONEY,POUNDS,BOW,CAPT
AI,BOAT,DOOR,DUKE,RING,MAP,FERRET
```

Ligne 2000

Il s'agit d'une ligne d'initialisation. Le nombre de verbes anglais et français est le même (34). Par contre, il y aura 35 noms français et seulement 18 noms anglais. Nous commencerons donc par initialiser

la variable NV à 34, puis, si nous nous trouvons en France (C33 % = 1), la variable NV sera égale à 35 et la lecture des DATA commencera à la ligne 2020 (RESTORE 2020). Sinon, la variable NN sera égale à 18 et la lecture des DATA commencera à la ligne 2040.

Ligne 2010

C'est dans cette ligne que se fait le remplissage des tableaux. Nous y trouvons une première boucle remplissant le tableau des verbes (VE$(I)) depuis le premier mot jusqu'au 34e. Une deuxième boucle suit immédiatement la première et remplit le tableau des noms (NO$(I)) depuis le premier jusqu'au NN^{ième}. Enfin, une fois que les dictionnaires sont chargés, nous terminons la ligne par un branchement au sous-programme de description des lieux.

Lignes 2020 à 2050

Ce sont les lignes de DATA. Il s'agit respectivement des verbes et des noms français puis des verbes et des noms anglais. L'instruction READ lisant *tous* les caractères compris entre deux virgules, il est important de noter qu'il ne faut pas laisser d'espaces entre les virgules et les mots (exception faite pour le premier de la liste, qui est obligatoirement séparé du mot DATA par un espace, et pour le dernier, qui est immédiatement suivi d'un <return>). Si cette règle n'était pas respectée, les mots contenus dans les dictionnaires seraient constitués d'un ou deux caractères de plus (les espaces ajoutés) que ceux proposés par le joueur et aucune occurrence ne pourrait être trouvée par l'analyse de syntaxe.

■ DIRECTIONS

TESTS INITIAUX

```
500 IF R$<>"N" AND R$<>"S" AND R$<>"E" A
ND R$<>"O" AND R$<>"W" OR R$="O" AND C33
%=0 OR R$="W" AND C33%=1 THEN 860
510 IF C>16 AND C<31 THEN IF C33%=1 THEN
 PRINT "VOUS ETES A L'INTERIEUR D'UNE PI
ECE":GOTO 10 ELSE PRINT "YOU ARE INSIDE
A ROOM":GOTO 10
520 IF C>9 AND C<17 AND C<>14 OR C>33 TH
EN GOSUB 810:IF K=0 THEN 10
```

Ligne 500

Si le mot contenu dans R$ est différent des cinq initiales de déplacement prévues OU que le mot est O alors que nous sommes en Angleterre OU que le mot est W et que nous nous trouvons en France, alors on va en 860 pour afficher un message, puis on retourne à la saisie.

Ligne 510

Si le numéro du lieu indique que l'on est à l'intérieur d'une pièce, on affiche un message puis on retourne à la saisie.

Ligne 520

Si l'on se trouve dans des lieux dans lesquels on ne peut se déplacer qu'à cheval, on va tester (GOSUB 810) si l'on est à pied. Si oui, on affiche un message puis on retourne à la saisie.

BRANCHEMENTS

```
530 ON C GOTO 540,550,570,580,590,590,60
0,570,570,610,620,630,640,580,650,660,51
0,510,510,510,510,510,510,510,510,510,51
0,510,510,510,670,700,720,730,740,750,76
0,780
```

Étant donné que, dans la majorité des cas, chaque lieu fait l'objet de déplacements (ou de conditions sur les déplacements) particuliers, chacun des lieux sera traité séparément dans des modules différents. Donc, en fonction du numéro du lieu (C), nous nous branchons au module concerné.

Chaque position de branchement derrière le GOTO doit correspondre au numéro du lieu : 540 correspond au lieu 1, 550 au lieu 2, 570 au lieu 3, etc. Autrement dit, il doit y avoir autant de branchements qu'il y a de lieux. C'est pourquoi, bien que nous ayons testé l'intérieur des pièces à la ligne 510, par sécurité nous remettons ce branchement dans la ligne 530 autant de fois qu'il est nécessaire.

TRAITEMENT DES DIRECTIONS

```
780 IF R$="N" THEN C=33:GOTO 1000
790 PRINT "PAS PAR LA : IL FAUT BIEN DES
 LIMITES   AU JEU !":GOTO 10
```

Il s'agit du principe général : nous avons choisi le 38e branchement de la ligne 530, nous nous trouvons donc dans le lieu n° 38 (devant la Jument bleue). Le seul déplacement possible est vers le nord. Si R$ contient la lettre N, alors on met dans C le numéro du lieu de destination, c'est-à-dire celui de la campagne (33), puis on effectue le branchement vers le module de description du lieu. Si R$ contient autre chose que N, la ligne 780 est sautée et la ligne 790 exécutée.

```
550 IF R$="E" THEN GOSUB 810:IF K=1 THEN
 C=33:GOTO 1000 ELSE 10
560 IF R$="O" THEN C=7:GOTO 1000 ELSE 79
0
```

Nous sommes devant la maison de la veuve (2e branchement, C = 2). Le déplacement vers l'est n'est permis que si l'on est à cheval. Par contre, le déplacement vers l'ouest est libre. Enfin, il n'y a aucun déplacement possible vers le nord ou vers le sud.

Ligne 550

Si R$ contient la lettre E, on se branche au sous-programme 810 (test à pied ou à cheval) puis, dans le cas où l'on est à cheval, on met le numéro du lieu de destination dans C et on va à la description du lieu. Sinon, si l'on est à pied, on retourne à la saisie.

Le traitement de la ligne 560 est tout à fait identique à celui présenté à la ligne 780.

```
570 IF C=3 AND R$="O" OR C=9 AND R$="S"
OR C=8 AND R$="E" THEN C=31:GOTO 1000 EL
SE 790
```

Lorsque nous nous trouvons dans certains lieux (comme par exemple devant les cuisines, devant la salle des gardes et devant le boudoir de la reine), le lieu de destination est le même (cour de Versailles, dans notre cas). Il est donc intéressant, pour des raisons d'économie de mémoire, d'effectuer les trois tests dans une même ligne de programme, en les groupant à l'aide de OR et de AND.

La compréhension de la ligne 570 n'offre aucune difficulté puisque, à part le regroupement des tests, nous utilisons le même principe de base que précédemment.

Nous avons choisi les trois exemples les plus représentatifs. Il est inutile de les passer tous en revue puisqu'ils sont tous basés sur le même principe.

■ MESSAGES COMMUNS

Il existe deux types de messages affichés par le programme en réponse à une action décidée par le joueur :

☐ *Les messages particuliers :* ils ne se rapportent qu'à une situation particulière et une seule, à un endroit précis du programme (description du lieu, par exemple).

☐ *Les messages communs :* ce sont des messages qui peuvent s'appliquer à plusieurs situations. Ils seront donc constitués d'une seule ligne de programme qui pourra être appelée à partir de plusieurs endroits. Là encore, ils sont de deux types : les messages simples et les messages appliqués. Prenons deux exemples.

1. Dans notre programme, il existe plusieurs lieux dans lesquels le joueur peut payer ou acheter quelque chose. Si le joueur n'a plus l'argent nécessaire, la phrase "VOUS N'AVEZ PLUS ASSEZ D'ARGENT" sera affichée en message *simple* (voir ligne 16060).
2. Dans la majorité des lieux, il existe des actions interdites. Chaque action est représentée par un verbe (stocké dans V$ à la ligne 115). Les messages *appliqués* (voir ligne 4100) seront donc constitués d'une partie invariable ("VOUS NE POUVEZ PAS") à laquelle sera ajoutée une partie appliquée à la situation (impression du verbe V$). Bien entendu, ces deux types de messages communs pourront être appelés en fonction des besoins à partir de n'importe quel endroit du programme.

Il est évident que si le concepteur ne prévoit, dans son logiciel, que des messages communs, le joueur éprouvera très vite une certaine lassitude ou un certain énervement qui se traduira par un désintéressement plus ou moins rapide. Par contre, un logiciel ne comportant que des messages particuliers sera idéal au point de vue de l'intérêt du jeu, mais conduira vite à une place de stockage en mémoire prohibitive. Le concepteur doit donc s'attacher à un bon compromis entre les deux types de messages.

■ SAUVEGARDE DE LA PARTIE EN COURS

Rien n'est plus crispant, lorsque l'on a souffert comme un damné pour en arriver presque au but final, de tomber sur un ultime piège

qui nous envoie *illico presto ad patres*, et *manu militari* de surcroît ! On peut imaginer combien il est désespérant de reprendre la partie à zéro et de reparcourir le trajet complet pour revenir à l'instant critique. Il est donc fort utile de prévoir un sous-programme permettant, à tout instant, de sauver sur la disquette de jeu l'ensemble du trajet déjà effectué par le joueur, puis de le restituer à la demande.

Pour cela, il suffira de sauvegarder uniquement le tableau des conditions tel qu'il se présente à ce moment-là. En effet, c'est lui et lui seul qui contient toutes les informations nécessaires au déroulement du jeu.

```
17000 GOSUB 17700:IF V8=34 THEN 17500
17300 OPENOUT "J":PRINT £9,EN:PRINT £9,C
0%,C1%,C2%,C3%,C4%,C5%,C6%,C7%,C8%,C9%,C
10%,C11%,C12%,C13%,C14%,C15%,C16%,C17%,C
18%,C19%,C20%,C21%,C22%,C23%,C24%,C,C26%
,C27%,C28%,C29%,C30%,C31%,C32%,C33%,C34%
,C35%,C36%,C37%,C38%:CLOSEOUT:GOTO 10
17500 OPENIN "J":INPUT £9,EN:INPUT £9,C0
%,C1%,C2%,C3%,C4%,C5%,C6%,C7%,C8%,C9%,C1
0%,C11%,C12%,C13%,C14%,C15%,C16%,C17%,C1
8%,C19%,C20%,C21%,C22%,C23%,C24%,C,C26%,
C27%,C28%,C29%,C30%,C31%,C32%,C33%,C34%,
C35%,C36%,C37%,C38%:CLOSEIN:GOTO 2000
17700 CLS:PRINT "METTEZ FACE 1":WHILE IN
KEY$="":WEND:CLS:LOAD "D":IF PEEK(65529)
<>1 THEN 17700 ELSE RETURN
```

D'une manière générale le sous-programme appelé par le verbe *sauver* est situé en 17300 et le sous-programme appelé par le verbe *reprendre* se trouve en 17500.

Ligne 17000

Elle commence par un branchement vers un sous-programme situé en 17700. Au retour de ce sous-programme, un test est effectué sur le numéro du verbe. Si celui-ci est *reprendre* (34), le branchement s'effectue à la ligne 17500. Sinon (si le verbe est *sauver*), la ligne suivante est exécutée.

Ligne 17300

On commence par ouvrir un fichier de sortie qui s'appellera J. Puis nous donnons l'ordre à la machine d'assigner l'unité de disquette (# 9) en lui indiquant le nombre d'enregistrements qu'il y aura à transmettre (variable EN que nous avons initialisée à 39 en début de programme). Enfin, nous enregistrons sur le disque toutes les variables de conditions avant de fermer le fichier et de retourner à la saisie située en ligne 10.

Ligne 17500

Il s'agit donc des instructions de chargement des variables de conditions en mémoire à partir du disque. Il utilise le même principe que pour la ligne précédente. Cependant, à la fin de la ligne, nous retournons au programme de chargement du dictionnaire correspondant au lieu indiqué par la variable C33 % que nous venons de charger.

Ligne 17700

Étant donné que nous avons décidé de sauver les variables de conditions sur la face 1 de la première disquette, ce petit sous-programme teste simplement si la bonne face est introduite. Ce principe sera expliqué dans la deuxième partie de cet ouvrage.

6. AIDE A LA MISE AU POINT

Lors de la conception et de la mise au point d'un logiciel de jeu d'aventure, il est difficile de tout prévoir et notamment les "bugs" qui risquent de "planter" le programme. La mise au point est une opération longue et fastidieuse. En effet, il faut parcourir le jeu dans tous ses recoins pour tester toutes les conditions et toutes les possibilités prévues ou non prévues. Dans une des versions précédentes, obnubilés par les pièges que nous pourrions tendre à l'intérieur de l'estaminet, nous avions complètement oublié le piège à effet retardé. Ainsi, le fait de voir ou de ne pas voir les mousquetaires ne servait à rien pour la poursuite du jeu. L'ami auquel nous avions confié les disquettes aux fins de test a eu tôt fait de découvrir ce "détail". D'autre part, personne n'est à l'abri d'une faute de frappe, surtout avec un listing aussi long. Au cours du test, lorsqu'une erreur est détectée par le programme, il est nécessaire d'en sortir, donc d'abandonner le jeu pour pouvoir la corriger. Il est alors fastidieux, surtout si l'erreur s'est produite vers la fin, de reprendre le jeu depuis le début et de reparcourir l'ensemble des lieux pour arriver à l'endroit incriminé avec les bons objets et les bonnes conditions.

Pour ces raisons, nous avons inclus, dans le programme en cours d'élaboration, un sous-programme qui permet de changer la valeur des conditions, à n'importe quel moment et à partir de n'importe où. On pourra ainsi, par exemple, quel que soit le lieu où l'on se trouve, aller directement devant la Jument bleue (C = 38) pour tester ce qu'il s'y passe en fonction du cheval possédé (C11 % = 0 à 6) sans être obligé de retourner à l'écurie ou dans la cour de Buckingham pour y changer de bestiole.

Bien entendu, ce sous-programme sera détruit dans la version définitive du jeu.

Afin de faciliter les différentes visualisations ou modifications des variables conditions, nous utiliserons la méthode d'itération, en incluant le traitement dans une boucle agissant sur un tableau. C'est la raison pour laquelle, dans la phrase de conception, lorsque nous n'avions pas encore de problèmes de place mémoire, les conditions étaient inscrites dans un tableau de variables réelles indicées. Lorsque le programme a été terminé et testé, et qu'il a fallu y inclure la partie graphique, nous avons buté sur le fameux message MEMORY FULL ! Il a donc fallu "tailler dans la masse" et trouver des solutions. Une de celles-ci, étant donné l'utilisation intensive que nous faisions de ces conditions, était de modifier la manière de les stocker. C'est pourquoi, dans le programme définitif, elles ont toutes été transformées en variables entières, donc non indicées. En voici la raison : outre les premiers octets indiquant

le type et le nom de la variable, le stockage de la valeur d'une variable simple entière ne prendra que 2 octets. Par contre, pour enregistrer la valeur d'une variable réelle indicée, 9 octets au minimum seront nécessaires.

Dans un but de simplification, la transformation des variables réelles indicées en variables entières s'effectuera selon le principe suivant : à part la variable C(25) qui devient simplement C, les autres conditions gardent le même numéro (C(1) devient C1 %, C(2) devient C2 %, etc.). En ce qui concerne la variable C, il est plus intéressant de la conserver sous la forme de variable réelle (non indicée, bien sûr). En effet, la perte de place mémoire qui résulte de son stockage est largement compensée par l'économie d'octets réalisée en n'inscrivant que la lettre C (occupant 1 octet) au lieu de C25 % (occupant 4 octets). Cela n'est valable que pour cette variable, car elle est inscrite un nombre important de fois dans le programme BASIC.

Puisqu'il s'agit d'un programme de mise au point, et afin de simplifier le chargement ou la lecture des variables de conditions à l'intérieur de boucles, le programme décrit ci-après utilise les variables réelles indicées. Les fonctions principales de cet utilitaire seront les suivantes :

- Affichage des conditions, en clair et sous forme de tableau, avec, pour chacune d'elles, l'affichage de la valeur qu'elles contiennent.
- Demande de changement des conditions (numéro puis valeur), une par une, et par rebouclage systématique.
- Affichage immédiat de la nouvelle valeur de la condition modifiée.
- Sortie du rebouclage en indiquant une valeur code (99) lors de la demande du numéro de condition.
- Après sortie du sous-programme, le branchement devra s'effectuer vers le sous-programme de chargement du dictionnaire. En effet, il se peut que le lieu de destination que l'on aura indiqué dans la variable correspondante (C(25)) soit situé dans un pays différent de celui dans lequel on se trouve.

```
10 IF R$="C" THEN 30000
```

Cette ligne doit être incluse dans l'analyse de syntaxe, dans la partie "traitement des initiales" située juste après la saisie de la réponse. Si

R$ contient la lettre C (comme <C>ondition), on effectue un branchement direct vers le sous-programme de changement situé à partir de la ligne 30000.

```
30000 CLS:RESTORE 30200:FOR I=1 TO 20:RE
AD CO$:LOCATE 1,I:PRINT CO$;C(I-1):NEXT
I
30010 FOR I=1 TO 19:READ CO$:LOCATE 22,I
:PRINT CO$;C(I+19):NEXT I
30050 LOCATE 1,23:PRINT SPACE$(79):LOCAT
E 13,23:INPUT "NO CONDITION : ",N:IF N=9
9 THEN CLS:LOCATE 1,25:GOTO 2000 ELSE LO
CATE 19,24:INPUT "VALEUR : ",V
30060 C(N)=V:IF N<20 THEN LOCATE 17,N+1:
ELSE LOCATE 37,N-19
30070 PRINT V:GOTO 30050
30200 DATA 0  NBRE ACTIONS=,1  VOIR BURE
AU =,2  AVOIR CONVOC=,3  AVOIR SAUF-C=,4
  OUVR TIROIR =,5  AVOIR BIJOUX=,6  OUVR
. ARMOIR=,7  AVOIR CAPE  =,8  OUVR. COFF
RE=,9  AV. CHAPEAU =,10 AVOIR  EPEE =,11
 NUM. CHEVAL =,12 AV. FERRETS =,13 AV. P
ISTOLE =
30210 DATA 14 CH. PISTOLE =,15 AV.  LIVR
ES =,16 VU MOUSQUET =,17 APPE PATRON =,1
8 PAYE CONSOM =,19 CHEVAL  1   =,20 CHEV
AL 2   =,21 CHEVAL 3   =,22 CHEVAL 4   =
,23 MONTR BAGUE=,24 AVOIR PLAN =,25 NUME
RO LIEU=
30220 DATA 26 A CHEVAL   =,27 AVOIR DORM
I=,28 AVOIR MANGE=,29 AV.P.SEJOUR=,30 FR
AP. PORTE=,31 FAIT REVER.=,32 AVOIR BAGU
E=,33 ETRE FRANCE=,34 OUVR. PORTE=,35 AP
P. CAPIT.=,36 CUEIL.FLEUR=,37 MONT.CONVO
C=,38 DONNE FLEUR=
```

Ligne 30000

Après le nettoyage de l'écran et l'initialisation du pointeur de DATA sur la première valeur à lire (ligne 30200), on trouve une première

boucle affichant les 20 premières conditions en colonne, sur la partie gauche de l'écran. Pour chaque valeur de I, on charge le DATA correspondant dans la variable CO$ que l'on imprime immédiatement à la colonne 1, ligne I, suivi de la valeur concernée (C(I-1) : C(0) pour I = 1, C(1) pour I = 2, etc.).

Ligne 30010

On utilise exactement le même principe pour afficher les 19 conditions restantes sur la partie droite de l'écran et en regard des 20 premières.

Ligne 30050

Sous le tableau affiché, à la ligne 23, on commence par effacer le message précédent éventuel, en imprimant tout simplement une série de 79 espaces (et non pas 80 pour éviter le retour automatique du curseur à la ligne suivante). Puis on affiche la demande du numéro de condition à modifier (N). Si la valeur indiquée est 99 (fin des modifications), on retourne directement au sous-programme de chargement du dictionnaire après avoir effacé l'écran. Sinon (si l'on indique un autre chiffre que 99), on demande la valeur correspondante que l'on veut affecter à la variable indiquée (V).

Ligne 30060

On commence par stocker la valeur V indiquée dans la variable considérée C(N). Puis on positionne le curseur à l'endroit où il faudra inscrire la valeur modifiée. Si, par exemple, nous avons indiqué que la condition 8 devait être modifiée (ce chiffre est inférieur à 20), le curseur sera positionné en colonne 17 (juste à droite de la première liste), ligne 8 + 1 = 9 (ligne où se trouve inscrite la condition n° 8 correspondante). De la même manière, si le numéro de condition à modifier est supérieur à 20, on positionne le curseur en colonne 37 (juste à droite de la deuxième liste), ligne N – 19 (correspondant au numéro de condition choisi).

Ligne 30070

Le curseur ayant été préalablement positionné, il ne reste plus qu'à afficher la nouvelle valeur de la condition, puis à se brancher au sous-programme de chargement du dictionnaire.

Lignes 30200 et 30210

Il s'agit là des lignes de DATA dans lesquelles seront indiquées les conditions en clair. Remarquez les espaces variables entre les mots ou les chiffres situés entre deux virgules : ils sont positionnés pour aligner toutes les conditions lorsqu'elles seront affichées à l'écran.

7. LE SON SUR AMSTRAD

Il n'est pas question, dans ce chapitre, de faire un cours magistral sur la théorie du son ou de la musique. Il existe par ailleurs une quantité d'ouvrages spécialisés traitant le sujet avec compétence et à tous les niveaux.

Si une mélodie n'est pas strictement obligatoire lors de la construction ou simplement à la résolution d'un jeu d'aventure, elle est cependant nécessaire, au même titre que le graphisme, pour en faire un produit fini donc agréable à utiliser.

L'objet de ce chapitre est, d'une part, de reprendre certains points qui nous ont semblé obscurs (voire erronés) afin de les éclaircir ou de les corriger et, d'autre part, de préciser certaines notions élémentaires de musique. Tout d'abord, nous vous proposons de remplacer le paragraphe 6.2 de la page 6.2 de votre manuel du 464 par le suivant :

■ QU'EST-CE QU'UN SON, UNE NOTE ?

Un son est l'effet que produisent les variations de pression du milieu ambiant (l'air, en général) sur l'oreille. Plus ces variations sont rapides (c'est-à-dire plus la fréquence est élevée), plus le son sera aigu.

Plus la différence de pression entre un maximum et un minimum (amplitude) est grande, plus le son sera fort.

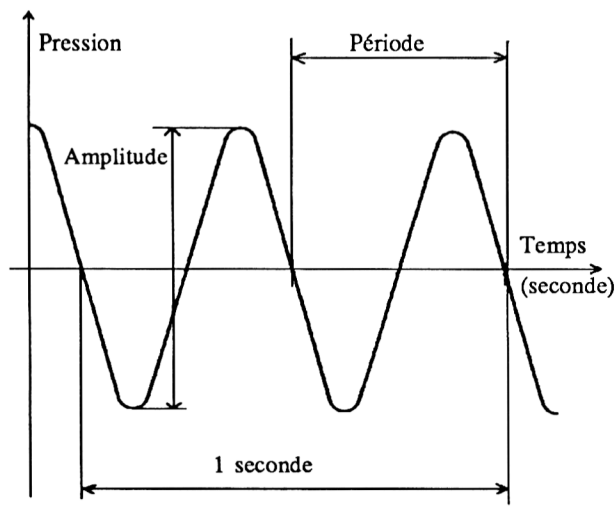


Figure 7 : Représentation physique d'un son

Un son produit par votre Amstrad sur l'un de ses trois canaux est constitué d'une suite d'oscillations qui se répètent pendant la durée du son.

La période

C'est la durée d'une oscillation, l'unité de temps étant la seconde.

La fréquence

C'est le nombre d'oscillations (ou le nombre de périodes) que l'on peut compter pendant une seconde. L'unité est le hertz (Hz). La Figure 7 représente un son de 2 oscillations par seconde (2 Hz).

Une note de musique

C'est un son dont la fréquence est musicalement et physiquement bien définie. Par convention internationale, la note de référence est le *la* dont la fréquence est actuellement fixée à 440 Hz.

Un intervalle

En physique des sons, un intervalle est la "distance" séparant deux notes de musique, mesurée en terme de rapport de fréquences. Si nous prenons l'exemple du *do* 32,703 Hz et du *fa* 43,654 Hz, l'intervalle séparant ces deux notes est de 43,654/32,703 = 1,33.

En musique (pour mémoire, et en simplifiant grandement), les intervalles portent des noms particuliers : un *intervalle de tierce* (ou *tierce*) est le nom de l'intervalle séparant, par exemple, un *do* du *mi* immédiatement supérieur, ou un *la* d'un *si*, ou encore un *ré* d'un *fa* dièse, etc. Il existe aussi des intervalles de seconde (*do* –> *ré* par exemple), de quarte (*do* –> *fa*), de quinte (*do* –> *sol*), de sixte (*do* –> *la*), et de septième (*do* –> *si* bémol).

Pour les mathématiciens curieux possédant une certaine culture musicale, sachez que pour un instrument tempéré, le rapport de fréquence entre deux notes consécutives séparées par un demi-ton est égal à $\sqrt[12]{2}$. A l'aide de cette formule et en partant de la fréquence du *la* international, on établit le tableau de fréquence de la page A 7.1 du manuel du 464.

Une octave

(Eh oui ! c'est bien un nom féminin !) C'est l'intervalle particulier séparant deux notes portant le même nom : *do* –> *do*, *ré* –> *ré*, etc. Le rapport de fréquence entre ces deux notes est égal à 2. Cela veut

dire qu'il faut multiplier la fréquence d'une note par 2 pour obtenir la fréquence de la même note située une octave au-dessus.

Du seul point de vue mathématique et physique, la fréquence et la période sont liées par la relation :

$$\text{Période (en secondes)} = \frac{1}{\text{Fréquence (en hertz)}}$$

Pour des raisons purement électroniques et informatiques qu'il est inutile d'expliquer ici, les concepteurs de l'Amstrad ont défini une "période" adaptée à leurs besoins. Pour cet ordinateur, la relation période/fréquence est la suivante :

$$\text{Période (spéciale Amstrad)} = \frac{62\ 500}{\text{Fréquence (en hertz)}}$$

Les sceptiques qui se sont précipités sur leur calculette afin de vérifier la validité de la formule ci-dessus ont dû être déçus ! Nous confirmons cependant que c'est bien la formule indiquée dans le manuel (et donc le tableau final) qui est fausse ! Au *la* 440 Hz correspond une période de 142. Au *do* 261,626 Hz correspond une période de 239. Il y a donc un décalage d'une octave entre les valeurs indiquées dans le tableau et la réalité. Si ce "détail" n'est pas trop gênant pour la programmation d'une mélodie de type familial, cela aurait pu agacer une oreille musicienne.

Il est important de noter que *plus la fréquence augmente,* plus le son devient aigu et *plus la période diminue.*

■ NOTIONS DE PROGRAMMATION DES SONS

Afin de ne pas surcharger cet ouvrage, centré sur la technique de réalisation d'un jeu d'aventure, nous réduirons notre étude sur la programmation des sons au mode d'emploi de l'instruction SOUND et des instructions directement associées (ENV, ENT). Nous ne parlerons que très peu de la technique des rendez-vous, en oubliant les queues et les interruptions.

Comme vous pourrez vous en rendre compte soit en vous servant des disquettes associées à cet ouvrage, soit en introduisant directement les morceaux de programme listés plus loin, cette restriction ne vous empêchera pas, cependant, d'obtenir des résultats des plus intéressants.

INSTRUCTION SOUND

Elle permet de jouer un son et un seul. Il faudra donc utiliser une instruction SOUND par note à jouer. Elle comporte, dans l'ordre, et séparés par des virgules, les paramètres suivants :

Canal

L'Amstrad possède trois canaux différents appelés A, B et C. C'est-à-dire qu'il peut jouer trois airs différents en même temps (un par canal, comme s'il y avait trois instruments). Cela permet, par exemple, de jouer la mélodie sur le canal A, l'accompagnement sur le canal B et la rythmique sur le canal C. Ce paramètre est constitué d'un nombre compris entre 0 et 255. Comme nous ne parlerons que très fugitivement des rendez-vous entre canaux, nous nous limiterons aux nombres compris entre 0 et 7.

La mention de cette valeur est obligatoire. Elle est destinée à aiguiller le son sur un ou plusieurs canaux, selon le code suivant :

0 Le son n'est pas joué (aucun canal n'est spécifié).
1 Le son est dirigé uniquement sur le canal A.
2 Le son est dirigé uniquement sur le canal B.
4 Le son est dirigé uniquement sur le canal C.
3 (1 + 2), le son est dirigé à la fois sur les canaux A et B.
5 (1 + 4), le son est dirigé à la fois sur les canaux A et C.
6 (2 + 4), le son est dirigé à la fois sur les canaux B et C.
7 (1 + 2 + 4), le son est dirigé à la fois sur les canaux A, B et C.

Période

Sa mention est obligatoire. Il s'agit de la hauteur du son et elle est indiquée dans le tableau de la page A 7.1 du manuel du 464 (avec le décalage d'une octave, comme nous l'avons souligné dans le paragraphe précédent). C'est un nombre théoriquement compris entre 16 (*si* de l'octave 5) et 3822 (*do* de l'octave – 2). N'oubliez pas que plus ce nombre est élevé, plus le son est grave (et vice versa). Le chiffre 0 peut être indiqué lorsque aucun son ne doit être entendu. Comme nous le verrons plus loin, cette particularité (qui peut paraître bizarre, de prime abord) nous sera cependant utile pour se passer de la technique

de rendez-vous. Enfin, rien n'interdit, au contraire, d'utiliser des chiffres non mentionnés dans le tableau. Par exemple, 2500 produira un son qui sera compris entre le *sol* et le *sol* dièse de l'octave – 2.

Durée

Sa mention est facultative. La durée élémentaire d'un son (le plus court qui puisse être produit) est fixée à 0,01 seconde. Le chiffre qui sera indiqué pour ce paramètre, multiplié par la durée élémentaire, représentera la durée totale du son en secondes. Ainsi, le chiffre 100 produira un son d'une durée de 100 x 0,01 = 1 seconde. Donc, si le chiffre 1 est indiqué, l'ordinateur jouera la note pendant la durée élémentaire (0,01). Lorsque ce paramètre n'est pas précisé, la durée du son est fixée, par conception, à 0,2 seconde par défaut.

Volume

Mention facultative. Normalement, c'est un nombre de 0 à 7, pouvant atteindre 0 à 15 si une enveloppe de volume est précisée dans la même instruction SOUND. C'est l'équivalent du bouton de volume d'un amplificateur. Plus le chiffre est élevé, plus le son est fort. Si on ne précise pas ce paramètre, l'ordinateur utilisera de lui-même le chiffre 4.

Enveloppe de volume et de ton

Ce sont des paramètres facultatifs pouvant varier de 0 à 15. Il s'agit tout simplement du numéro de référence de l'enveloppe de volume ou de ton (voir ci-dessous ENV et ENT) que l'ordinateur doit utiliser pour jouer la note considérée. Par défaut, l'ordinateur utilise la valeur 0 (pas d'enveloppe).

Bruit

C'est un paramètre facultatif pouvant varier de 0 à 31 (et même plus pour certains effets spéciaux dont nous ne parlerons pas ici). Ce paramètre permet de superposer au son déterminé ce que les acousticiens appellent un *bruit blanc*. Il s'agit d'une sorte de souffle crachotant qui peut éventuellement servir pour imiter vaguement un tas de choses comme le bruit d'une cymbale ou, selon la durée et la hauteur du son qui lui est associé, le bruit d'une foule de spectateurs exprimant sa joie devant un but marqué. Par défaut, l'ordinateur utilise la valeur 0 (pas de bruit).

INSTRUCTION ENV

Lorsqu'une note est jouée sur un canal quelconque, sans mention d'enveloppe, le son apparaît brutalement avec une certaine puissance, garde cette puissance constante pendant toute la durée de la note, puis s'éteint aussi brutalement qu'il avait commencé. Cet effet désagréable peut être compensé en modulant l'attaque (montée plus ou moins progressive de la puissance du son), la fin (extinction progressive) et la puissance entre les deux.

Cela est réalisé par l'instruction ENV. Cette instruction étant bien expliquée page 6.4 du manuel du 464, nous n'y reviendrons pas. Précisons seulement que le tout premier paramètre (N) est le numéro de repère de l'enveloppe. Ce numéro doit être un nombre de 1 à 15, ce qui autorise au maximum la définition de 15 enveloppes différentes qui pourront être appelées, en fonction des besoins, à partir des instructions SOUND (une seule enveloppe de volume par instruction). Les autres paramètres sont groupés en cinq sections de trois, ce qui permet de programmer, pour une enveloppe, jusqu'à cinq variations de volume pour la même note. Dans la majorité des cas, trois suffiront : une attaque, un *sustain* (maintien du son après le début de l'extinction) et une section intermédiaire.

Plus la taille du pas sera grande, plus la variation de volume sera rapide. Plus le temps de pause sera grand, plus la durée du phénomène sera longue. Bien entendu, la hauteur du pas, multipliée par le nombre total de pas, doit être égale au volume indiqué dans l'instruction SOUND. De la même manière, le temps de pause, multiplié par le nombre total de pas, doit correspondre à la durée de cette note.

INSTRUCTION ENT

Elle a exactement la même structure et les mêmes limites que l'instruction ENV précédente (voir page 6.8 du manuel du 464). Au lieu de faire varier le volume sonore de la note, elle fait varier sa fréquence, ce qui donne un effet de *glissando* ou de *vibrato.*

En ce qui concerne la taille du pas, il faut savoir que, contrairement à ce que la figure du manuel peut le laisser penser, une *taille de pas positive* donne une *augmentation de la période*, donc une *diminution de la fréquence.* Cela veut dire que le son devient plus grave lorsque la taille du pas est positive, et inversement.

Pour chacune des deux instructions ENV et ENT, lorsqu'elles sont employées, le numéro d'enveloppe et une section complète (comprenant

ses trois paramètres) sont obligatoires. Les quatre autres sections sont facultatives.

Nous vous donnons ci-après quelques précisions complémentaires et importantes, qui ne figurent pas dans la notice ou dont les explications sont nébuleuses. Ces précisions sont volontairement partielles ou simplifiées par rapport aux possibilités de la machine. Ce faisant, nous pensons faciliter grandement la tâche des personnes essayant d'y voir clair afin de se lancer dans des réalisations musicales à caractère non professionnel.

Au-dessus du HIMEM, dans la partie réservée à l'interpréteur et au système (entre les adresses 42600 et 49152), l'Amstrad réserve une partie de mémoire RAM pour stocker les différentes instructions SOUND qu'il rencontre au cours du déroulement d'un programme. En première approximation, ces mémoires sont utilisées par l'ordinateur de la même manière que le terrain du jeu "puissance 4" qui ne comporterait que quatre cases sur trois.

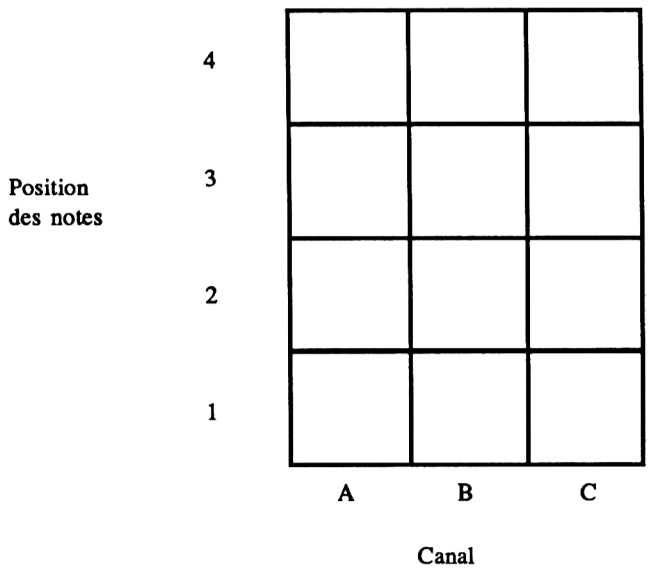


Figure 8

Prenons un exemple. Soit une ligne programme ainsi constituée :

SOUND 1,... : SOUND 1,... : SOUND 1,... : etc.

(1re note) (2e note) (3e note), etc.

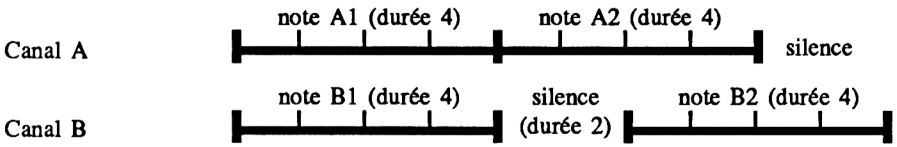
c'est-à-dire une mélodie jouée entièrement sur le canal A. L'ordinateur va lire les instructions les unes à la suite des autres, repérer chaque fois le numéro de canal, et mettre les paramètres correspondants dans la colonne indiquée par ce numéro. Dans notre cas, les notes seront stockées à la queue leu leu dans la colonne A (note 1 dans A1, note 2 dans A2, etc.) jusqu'à ce que l'ordinateur décide de les jouer. Au fur et à mesure que cette queue se libère, le processus continue jusqu'à la dernière instruction SOUND.

L'affaire se complique un peu lorsqu'on veut faire jouer une mélodie à deux ou trois voix (une par canal). Prenons l'exemple d'un air comportant trois lignes mélodiques différentes. Chaque ligne doit être jouée en même temps que les autres sur un canal différent. Dans ce cas, il est nécessaire de grouper les trois instructions SOUND correspondant à la première note de chaque ligne musicale, les unes à la suite des autres (SOUND 1,... : SOUND 2,... : SOUND 4,...), puis, à la suite, les trois instructions de la deuxième note de chacune des lignes, etc. Il ne faut surtout pas, à la limite, regrouper toutes les notes de la voie A, par exemple, puis, immédiatement à la suite, toutes les notes de la voie B, etc. En un mot, il faut obligatoirement écrire les instructions SOUND correspondant aux notes devant être jouées en même temps sur des canaux différents, les unes à la suite immédiate des autres, puis recommencer (éventuellement à la suite des précédentes) avec les notes suivantes.

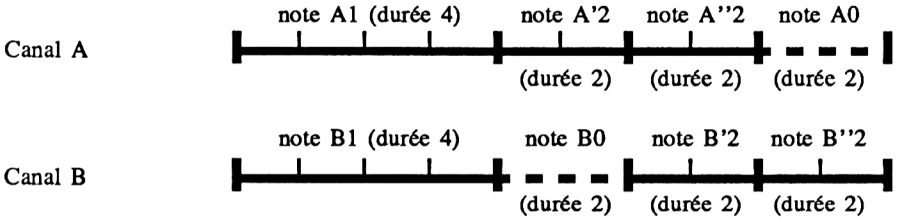
Mais, allez-vous penser à juste titre, il se peut que dans la progression de la ligne mélodique de la voie B, par exemple, il y ait un "trou" (un silence) en plein milieu, alors que sur les autres canaux des notes doivent être jouées. Dans notre exemple à trois voix, puisqu'il doit obligatoirement y avoir trois instructions SOUND groupées, il suffira d'y inclure l'instruction SOUND 2,0,... (SOUND 1,0,... pour la voie A, etc.). Étant donné qu'il y est indiqué une période égale à 0, la note sera présente, mais le son ne sera pas entendu !

La complication atteint le summum lorsqu'il est question de durée différente pour des notes jouées sur les trois canaux ! La Figure 9 vous aidera à comprendre ce qu'il faut faire dans ce cas.

Durée des notes à jouer



Décomposition des durées



Paramètres des instructions : SOUND

Note	Paramètres	Note	Paramètres
A1	SOUND 1,a1,4	A''2	SOUND 1,a''2,2
B1	SOUND 2,b1,4	B'2	SOUND 2,b'2,2
A'2	SOUND 1,a'2,2	A0	SOUND 1,0,2
B0	SOUND 2,0,2	B''2	SOUND 2,b''2,2

Ligne programme

```
10 SOUND 1a1,4 : SOUND 2,b1,4 : SOUND 1,a'2,2 : SOUND 2,0,2 :
   SOUND 1,a''2,2 : SOUND 2,b'2,2 : SOUND 1,0,2 : SOUND 2,b''2,2
```

Figure 9 : Programmation de notes sur 2 canaux différents

De toute façon, les exemples que nous allons traiter vous aideront à comprendre. Contrairement à ce qu'on pourrait penser, la programmation d'une mélodie n'est pas du tout compliquée. Il suffit simplement d'être très minutieux dans la réalisation du décompte des durées et du partage des notes selon le principe indiqué sur la Figure 9. Les durées indiquées tiennent compte de la durée totale d'une éventuelle enveloppe de volume ou de ton associée (attaque + sustain + variations intermédiaires).

■ APPLICATION A LA SONORISATION DU JEU

Nous avons prévu trois sortes de musiques différentes : une pour le début (qui nous servira aussi pour la fin en cas de réussite), une pour la transition France-Angleterre et une pour le cas où l'on a perdu. Bien entendu, il vous est possible de les changer ou d'en ajouter d'autres. Dans ce cas, il faudra faire attention à la place mémoire...

SONORISATION DE FIN

Pour le cas où l'on perdrait, nous allons programmer un son un peu bizarre, composé d'un long glissando descendant se terminant par un vibrato lent, le tout noyé dans un bruit blanc. Nous mettrons donc en application l'instruction ENT et le paramètre bruit de l'instruction SOUND.

```
50010 ENT 1,150,1,1:ENT 2,20,-6,1,20,6,1
,20,-6,1,20,6,1
50020 SOUND 1,60,150,7,,1,1:SOUND 2,67,1
50,7,,1:SOUND 1,210,150,7,,1,1:SOUND 2,2
17,150,7,,1:SOUND 1,360,100,7,,2,1:SOUND
 2,367,100,7,,2
```

Ligne 50010

Nous définissons deux enveloppes de ton qui nous serviront, l'une pour le glissando (ENT 1), l'autre pour le vibrato final (ENT 2).

Le glissando (SOUND 1) étant une variation continue et constante de la période, une seule section suffira. Comme nous le verrons plus loin, lors de l'étude de la ligne 50020, le paramètre durée de chaque

instruction SOUND est fixé à 150. D'autre part, pour que le son paraisse continu et relativement lent, après essais nous choisissons le temps de pause minimal, c'est-à-dire 1. Le paramètre nombre de pas sera donc égal à 150/1 = 150 (si le temps de pause avait été choisi à 2, nous aurions obtenu un nombre de pas de 150/1 = 75). Enfin, la taille du pas est fixée à 1.

Pour le vibrato final (ENT 2), nous souhaitons une variation en "dent de scie" de la fréquence deux fois de suite. Nous renseignons donc quatre sections de trois paramètres. La montée et la descente de chaque dent de scie est symétrique. Pour chaque section nous avons fixé, à l'oreille, un nombre de pas de 20, une taille de pas de – 6 pour la montée (+ 6 pour la descente) et un temps de pause de 1.

Ligne 50020

Une fois les enveloppes de ton fixées, il est nécessaire de donner à l'ordinateur l'ordre de jouer les sons. Vous remarquerez que nous avons programmé deux canaux (SOUND 1 et SOUND 2, correspondant aux canaux A et B). Pour l'instant, intéressons-nous aux instructions SOUND 1 (canal A).

Pour la première instruction, nous partons du *do* de l'octave 4, dont la période est de 60. La durée est fixée à 150, le son aura un volume de 7, nous ne voulons pas d'enveloppe de volume (nous aurions pu mettre 0, mais cela gagne de la place mémoire !), l'enveloppe de ton est l'enveloppe n° 1 (glissando) et enfin, comme nous voulons ajouter du bruit blanc pas trop fort, nous mettons 1 comme dernier paramètre.

Que va-t-il se passer ? Lorsqu'il va rencontrer la première instruction SOUND et son enveloppe ENT 1 associée, l'ordinateur va commencer par jouer la note de période 60. Puis, toutes les 0,1 seconde (puisque le temps de pause = 1), il va augmenter cette valeur de 1 (taille du pas = 1), et cela 150 fois de suite. Lorsqu'il aura fini, la période aura donc atteint la valeur 60 + 150 x 1 = 210 et le son s'arrêtera. Comme nous voulons continuer encore vers un son plus grave, il nous suffira d'ajouter une seconde instruction SOUND 1 avec 210 comme hauteur de note de départ et avec la même référence pour l'enveloppe de ton. La valeur finale de la période de ce second son atteindra 360.

Mais, pensez-vous, pourquoi avoir deux instructions SOUND alors qu'une seule suffirait, à condition de préciser, dans ENT correspondant, un nombre de pas de 300 ? Tout simplement parce que ce paramètre ne peut prendre de valeur supérieure à 239 et que nous avons besoin de 300. Nous avons donc partagé cette valeur exactement en deux, de manière que la même instruction ENT serve pour les deux SOUND.

Pour notre goût, le son défini précédemment produit un effet trop clair, trop conventionnel et pas assez bizarre. Afin de rendre l'effet voulu, nous ajoutons une deuxième série d'instructions jouant sur le canal B (SOUND 2), strictement identique, mais décalée d'un intervalle dissonant (un intervalle de tierce *do-mi* est consonant. Un intervalle de seconde *do-ré* ou *do-si* bémol est dissonant). La valeur de période de départ pour cette paire de SOUND 2 sera donc de 67 (*si* bémol). Bien entendu, les deux séries de SOUND devront être imbriquées, comme on peut le voir à la ligne 50020, pour que ces sons soient joués en synchronisme.

La dernière paire d'instructions SOUND 1 - SOUND 2 produira l'effet de vibrato final en prenant l'enveloppe de ton n° 2. La variation de ton se fait selon le même principe que précédemment et l'intervalle de seconde (dissonant) est conservé.

MUSIQUE DE TRANSITION

Puisqu'il s'agit de bateau et de mer, nous choisissons les premières mesures de l'air célèbre *Il était un petit navire* dont vous trouverez la partition sur la Figure 10.

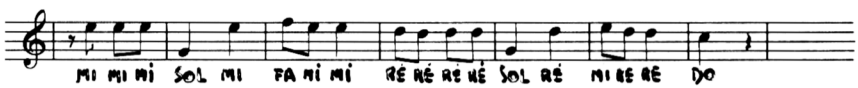


Figure 10 : Il était un petit navire.

Comme pour le son précédent, nous allons rendre cette mélodie surprenante et sortant de l'ordinaire par l'utilisation des intervalles dissonants et l'adjonction de bruit blanc. La mélodie principale sera jouée sur le canal A, tandis que la mélodie translatée sera jouée en synchronisme sur le canal B. Seul le canal A se verra affecter le bruit blanc.

Commençons par le plus facile : en partant du *mi* de l'octave 4, les périodes des différentes notes utilisées sont les suivantes (les notes devant être jouées en même temps se trouvent l'une au-dessus de l'autre) :

Canal A	MI = 47	SOL = 80	FA = 45	RE = 53	DO = 60
Canal B	DO = 56	MI = 95	RE = 53	SI = 63	LA = 71

Afin d'équilibrer la puissance sonore des deux mélodies, nous avons choisi un volume de 15 sur le canal A et de 3 sur le canal B. Le bruit ajouté sur le canal A sera de 1. Enfin, par expérience, ou à l'oreille à l'aide d'essais, nous déterminons une durée de 56 pour la noire (♩) donc de 56/2 = 28 pour la croche (♪). Le paramètre durée des instructions SOUND doit donc être de 56 lorsqu'elles jouent une noire et de 28 lorsqu'elles jouent une croche. En faisant la somme de tous les pas de chacune des éventuelles enveloppes de volume associées, on devrait trouver les mêmes chiffres.

Jouée telle quelle, sans enveloppe de volume, cette mélodie n'est pas parfaite. Il y manque un "piqué" sur certaines notes (notes dont la partie entendue est brève comparée à la durée totale). Nous pourrions utiliser le moyen indiqué sur la Figure 9 et diviser chaque note en "sous-notes" avec utilisation des SOUND 1,0. En fait, puisque la même mélodie est jouée sur les deux canaux, nous allons utiliser une deuxième méthode qui consiste à définir une enveloppe de volume spécialement adaptée. Ce principe est illustré sur la Figure 11.

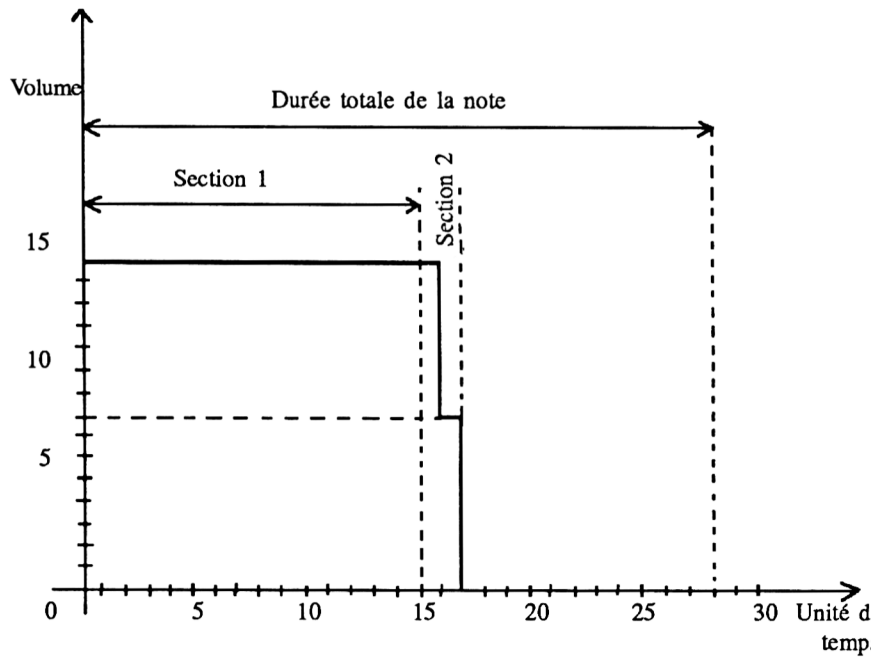


Figure 11 : Diagramme de l'instruction ENV 1.

La programmation d'un tel son sera donc :

ENV 1,1,0,15,2,– 7,2 : SOUND 1,47,28,15,1.

Dans ce cas, que va-t-il se passer ? La note va effectivement durer 28 unités de temps, comme indiqué dans l'instruction SOUND. Par contre, à cause des paramètres de l'enveloppe de volume, elle ne va être entendue que pendant 17 unités de temps (dont une à demi-puissance). Pendant les 11 unités de temps restant, le volume est à 0.

Le même principe est appliqué pour le canal B pour obtenir la ligne :

ENV 2,1,0,15,3,– 1,1 : SOUND 2,56,28,3,2.

```
45000 RESTORE 45010:ENV 1,1,0,15,2,-7,2:
ENV 2,1,0,15,3,-1,1:FOR I=1 TO 18:READ F
1,D,E,F2:SOUND 17,F1,D,15,E,,1:SOUND 10,
F2,0.8*D,3,2*E:NEXT:RETURN
45010 DATA 47,28,1,56,47,28,1,56,47,28,1
,56,80,56,0,95,47,56,0,56,45,28,0,53,47,
28,1,56,47,56,0,56,53,28,1,63,53,28,1,63
,53,28,1,63,53,28,1,63,80,56,0,95,53,56,
0,63,47,28,0,56,53,28,1,63,53,56,0,63,60
,28,0,71
```

Ligne 45000

Cette ligne débute par une instruction RESTORE indiquant au programme où il faut chercher les valeurs à charger (nous allons voir pourquoi un peu plus loin). Nous trouvons ensuite les deux enveloppes de volume définies précédemment.

Étant donné que nous avons 18 notes à programmer, il faudra donc inscrire 36 instructions SOUND complètes les unes à la suite des autres, ce qui tient une place rédhibitoire. Il est plus intéressant de les inclure dans une boucle comprenant les deux instructions SOUND 1 et SOUND 2 précédées par une instruction READ destinée à aller lire les paramètres variables des deux instructions. La borne supérieure de la boucle sera égale au nombre de notes à jouer.

Nous allons encore améliorer notre petite mélodie en raccourcissant la durée de toutes les instructions SOUND 2 de 20 % (0,8 x D). Jouée ainsi, sans autre précaution, cela donne un mélange cacophonique. En effet, les notes de la voie B n'ayant plus la même durée que celles

de la voie A, il arrive un moment où elles ont tendance à être jouées avant la note correspondante de la voie A. Pour remédier à cela, nous allons donner l'ordre à toutes les notes d'attendre que la plus lente ait fini de jouer pour démarrer en même temps qu'elle. Dans notre cas (où toutes les notes jouées sur le canal A possèdent une note correspondante sur la voie B), cela se réalise en donnant un numéro de canal particulier à chaque instruction SOUND : 17 = 1 (son envoyé sur la voie A) + 16 (rendez-vous avec la voie B) et 10 = 2 (son envoyé sur la voie B) + 8 (rendez-vous avec la voie A). Chaque fois, par exemple, qu'une note de la voie B est terminée, le chiffre 10 de l'instruction suivante concernant cette voie indique à l'ordinateur qu'il faut attendre l'apparition de la prochaine instruction de la voie A comportant le numéro de canal 17. Nous n'approfondirons pas plus cette technique.

Ligne 45010

Il s'agit de la ligne des DATA. Il y sera mentionné, les uns à la suite des autres et dans l'ordre strict dans lequel ils devront être lus, tous les paramètres à utiliser par les instructions SOUND de la boucle.

MUSIQUE DE DÉBUT

Nous avons gardé le meilleur pour la fin ! Vous trouverez en Figure 12 l'instrument de torture : une mélodie inédite dont la ligne mélodique (portée supérieure) sera jouée sur le canal A et l'accompagnement (portée du bas) sur le canal B. Contrairement à la mélodie précédente, on peut remarquer qu'entre la ligne mélodique et la ligne d'accompagnement les notes sont différentes tant en nombre qu'en durée.

Comme le choix des différents paramètres et leur programmation dans les instructions ressemble fort à ce que nous venons de voir, nous n'insisterons pas plus longtemps pour pouvoir nous consacrer au partage des temps, mesure par mesure.

Précisons simplement que puisque nous voulons une imitation de *piccolo* (flûte aiguë), nous choisirons une période de 24 pour la première note de la mélodie (*mi* de l'octave 5) et de 47 pour la première note de l'accompagnement (*mi* de l'octave 4). Les différentes périodes des notes suivantes s'en déduisent. Le tempo sera assez rapide : 28 unités de temps pour la noire (14 pour la croche).

Vous trouverez sur les Figures 13, 14 et 15 les diagrammes des

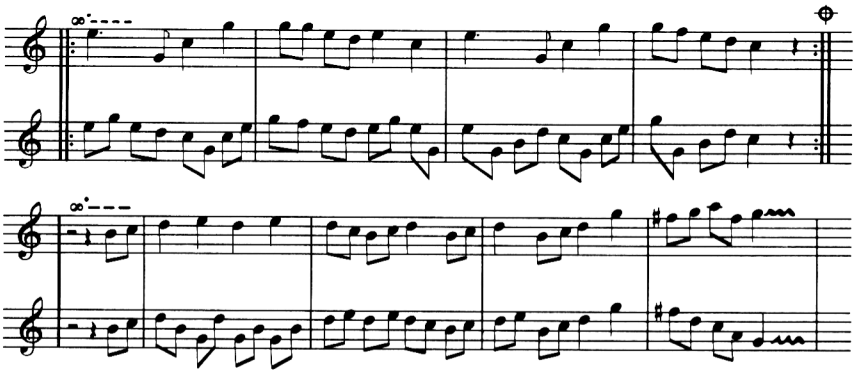


Figure 12 : Les Trois Mousquetaires.

durées des mesures représentatives prises comme exemple (mesures 1, 4 et 8).

Sur chaque figure on montre d'abord le diagramme des durées (voies A et B) de la mesure concernée, directement traduit de la partition. En dessous, on peut voir le diagramme modifié de façon qu'il y ait autant de notes de même durée sur la voie A que sur la voie B (le premier *mi* de la voie A pour la première mesure a été divisé en trois). Enfin, nous indiquons les instructions qu'il faut pour jouer cette mesure. Sur chaque diagramme, en dessous du nom de la note, figure sa période puis sa durée.

Cette méthode ne met pas en valeur toutes les possibilités de l'Amstrad dans ce domaine. L'explication complète de la technique des sons sur cet appareil dépasse largement le cadre de cet ouvrage. Elle a cependant le mérite d'être simple et rapide à mettre en œuvre. A partir d'une partition, elle permettra aux personnes n'ayant que des rudiments de solfège de programmer n'importe quelle mélodie complexe ou non, sur 1, 2 ou 3 canaux.

Nous pouvons maintenant examiner le programme complet de cette mélodie.

Les quatre premières mesures seront jouées deux fois de suite. Les quatre mesures suivantes ne seront jouées qu'une seule fois, puis nous terminerons en jouant encore deux fois les quatre premières mesures.

Bien entendu, s'il fallait inscrire, les unes à la suite des autres, toutes les instructions SOUND accompagnées de leurs paramètres, il en faudrait plus de 300 et la mémoire RAM perdrait quelque 7 ou 8K octets, au détriment du programme principal. Nous réduirons notablement cette place mémoire en incluant deux instructions SOUND dans une boucle.

Diagramme des durées partition

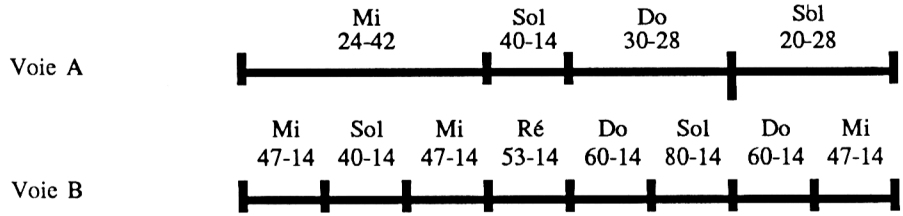
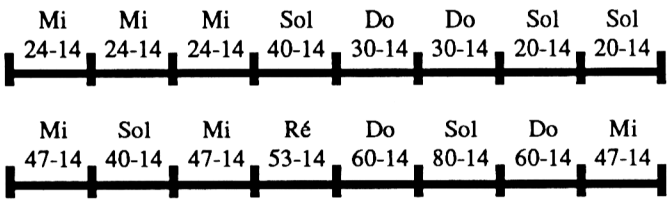


Diagramme des durées à programmer



Suite des instructions SOUND correspondantes

SOUND 1,24,14 : SOUND 2,47,14 : SOUND 1,24,14 : SOUND 2,40,14 : SOUND 1,24,14 : SOUND 2,47,14 : SOUND 1,40,14 : SOUND 2,53,14 : SOUND 1,30,14 : SOUND 2,60,14 : SOUND 1,30,14 : SOUND 2,80,14 : SOUND 1,20,14 : SOUND 2,60,14 : SOUND 1,20,14 : SOUND 2,47,14

Figure 13 : Diagramme des durées, première mesure

Diagramme des durées partition

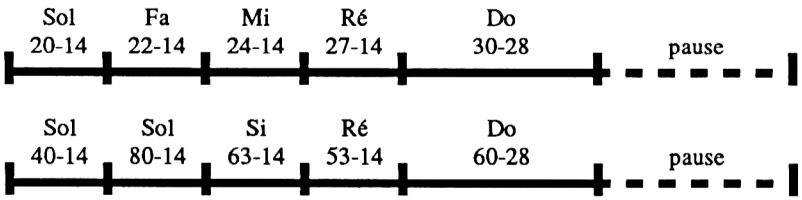
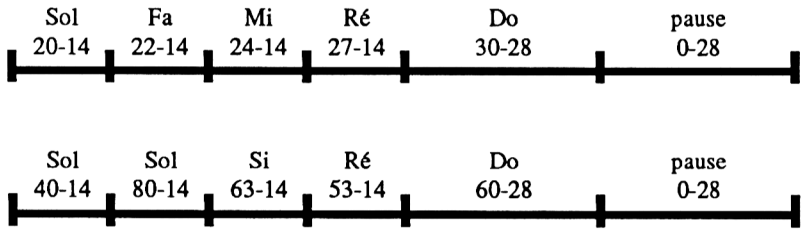


Diagramme des durées à programmer



Suite des instructions SOUND correspondantes

SOUND 1,20,14 : SOUND 2,40,14 : SOUND 1,22,14 : SOUND 2,80,14 : SOUND 1,24,14 : SOUND 2,63,14 : SOUND 1,27,14 : SOUND 2,53,14 : SOUND 1,30,28 : SOUND 2,60,28 : SOUND 1,0,28 : SOUND 2,0,28

Figure 14 : Diagramme des durées, quatrième mesure

Diagramme des durées partition

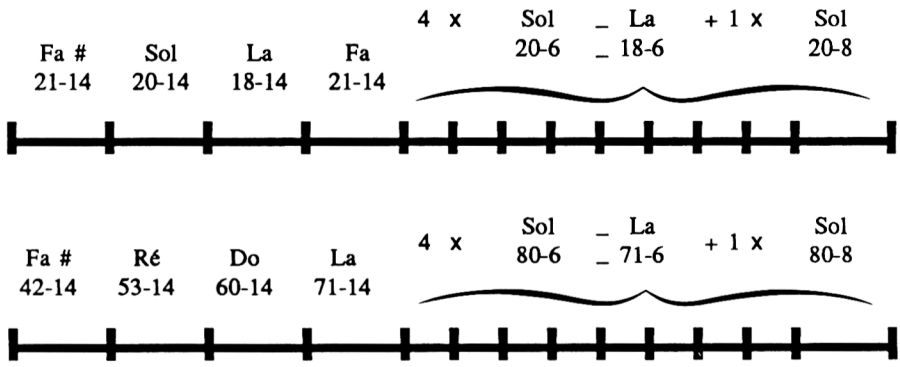


Diagramme des durées à programmer

Idem à la partition

Idem à la partition

Suite des instructions SOUND correspondantes

SOUND 1,21,14 : SOUND 2,42,14 : SOUND 1,20,14 : SOUND 2,53,14 : SOUND 1,18,14 : SOUND 2,60,14 : SOUND 1,21,14 : SOUND 2,71,14 : SOUND 1,20,6 : SOUND 2,80,6 : SOUND 1,18,6 : SOUND 2,71,6 : SOUND 1,20,6 : SOUND 2,80,6 : SOUND 1,18,6 : SOUND 2,71,6 : SOUND 1,20,6 : SOUND 2,80,6 : SOUND 1,18,6 : SOUND 2,71,6 : SOUND 1,20,6 : SOUND 2,80,6 : SOUND 1,18,6 : SOUND 2,71,6 : SOUND 1,20,8 : SOUND 2,80,8

Figure 15 : Diagramme des durées, huitième mesure

```
40000 RESTORE 40040:E=0:ENV 1,3,-5,8:ENV
 2,1,-15,0
40010 FOR I=1 TO 2
40015 READ F1,D1,E1,F2,D2:IF F1=-1 THEN
40030 ELSE IF F1=-2 THEN I=2:E=1:GOTO 40
030
40020 IF INKEY$<>"" THEN RETURN ELSE SOU
ND 1,F1,D1,15,E1:SOUND 2,F2,D2,4:GOTO 40
015
40030 RESTORE 40040:NEXT:IF F1=-2 THEN 4
0010 ELSE IF E=1 THEN RETURN ELSE RESTOR
E 40060:GOTO 40010
40040 DATA 24,14,0,47,14,24,14,0,40,14,2
4,14,0,47,14,40,14,0,53,14,30,14,1,60,14
,0,14,2,80,14,20,14,1,60,14,0,14,2,47,14
,20,14,1,40,14,22,14,1,45,14,24,14,1,47,
14,27,14,1,53,14,24,14,1,47,14,0,14,2,40
,14,30,14,1,47,14,0,14,2,80,14,24,14,0,4
7,14,24,14,0
40050 DATA 80,14,24,14,0,63,14,40,14,0,5
3,14,30,14,1,60,14,0,14,2,80,14,20,14,1,
60,14,0,14,2,47,14,20,14,1,40,14,22,14,1
,80,14,24,14,1,63,14,27,14,1,53,14,30,56
,1,60,56,-1,1,1,1,1
40060 DATA 32,14,0,63,14,30,14,0,60,14,2
7,14,1,53,14,0,14,2,63,14,24,14,1,80,14,
0,14,2,53,14,27,14,1,80,14,0,14,2,63,14,
24,14,1,80,14,0,14,2,63,14,27,14,1,53,14
,30,14,1,47,14,32,14,1,53,14,30,14,1,47,
14,27,14,1,53,14,0,14,2,60,14,32,14,0,32
,14,30,14,0,30
40070 DATA 14,27,28,1,27,28,32,14,0,63,1
4,30,14,0,60,14,27,28,1,53,28,20,28,1,40
,28,21,14,1,45,14,20,14,1,53,14,18,14,1,
60,14,21,14,1,71,14,20,6,0,80,6,18,6,0,7
1,6,20,6,0,80,6,18,6,0,71,6,20,6,0,80,6,
18,6,0,71,6,20,6,0,80,6,18,6,0,71,6,20,8
,0,80,8
40080 DATA -2,1,1,1,1
```

Les lignes de DATA 40040 et 40050 contiennent les paramètres des quatre premières mesures. Les lignes 40060 à 40080 contiennent les paramètres des quatre mesures suivantes. Notez les dernières valeurs des lignes 40050 et 40080 : ce sont des valeurs repères qui nous permettront de tester si la ligne mélodique est terminée.

Ligne 40000

On positionne le pointeur de DATA sur la première valeur de la ligne 40040. Puis on met le drapeau E à zéro et enfin, on définit deux enveloppes de volume.

Ligne 40010

C'est le début de la boucle qui permet de jouer deux fois les quatre premières mesures.

Ligne 40015

L'instruction READ lit les cinq premières valeurs de la ligne 40040 qui sont respectivement la période de la note du canal A (F1), la durée de cette note (D1), son numéro d'enveloppe (E1), la période de la note du canal B (F2) et sa durée (D2).

Si F1 est égal à la valeur repère – 1 (fin des quatre premières mesures), on se branche en ligne 40030 pour recommencer à jouer les quatre premières mesures.

Si F1 est égal à la valeur repère – 2 (fin des quatre mesures suivantes), nous mettons dans le compteur de boucle sa valeur finale pour éviter de rejouer une deuxième fois ces quatre mesures, puis on positionne E à 1 pour savoir qu'il faudra s'arrêter après avoir joué encore deux fois les quatre premières mesures. Enfin on se branche à la ligne 40030.

Ligne 40020

Afin de laisser la possibilité d'arrêter la musique à tout moment, on inclut l'instruction INKEY$. Lorsqu'une touche est frappée, on sort du sous-programme musical et on retourne au programme principal.

Sinon, on joue sur les deux canaux les deux instructions SOUND dont les paramètres F1, D1, E1, F2 et D2 viennent d'être lus puis on retourne en 40015 pour relire les cinq paramètres suivants.

Ligne 40030

Rappelons qu'on y vient à partir de la ligne 40015 si F1 = – 1 (fin des quatre premières mesures) ou si F2 = – 2 (fin des quatre mesures suivantes), c'est-à-dire lorsqu'il faut rejouer les quatre premières mesures, sauf si la fin de la boucle est atteinte.

On commence par positionner le pointeur de DATA sur la première valeur de la ligne 40040, puis on continue la boucle si elle n'est pas terminée.

Si F1 = – 2 (fin des mesures 4 à 8), on retourne au début de la boucle en 40010, pour jouer encore deux fois les quatre premières mesures. Une fois cela terminé, on trouve – 1 dans F1. Donc, arrivé au test IF F1 = – 2, celui-ci sera faux et on passera à la suite de la ligne.

Sinon, si E = 1 (on a déjà joué les mesures 4 à 8, sous-entendu : on vient de jouer deux fois les quatre premières mesures), alors on retourne au programme principal.

Sinon (si E est différent de 1) on n'a pas encore joué les mesures 4 à 8 et on retourne en 40010 pour le faire.

Si vous avez eu la patience de nous lire jusqu'à présent, nous vous incitons à continuer ! En effet, la deuxième partie de cet ouvrage est consacrée à la réalisation des images devant agrémenter le jeu.

Comme nous vous l'avons suggéré précédemment, un jeu d'aventure peut se passer d'images. Les seules indications sont alors fournies par le texte qui va défiler sur l'écran. Par expérience, et en tant qu'utilisateurs, nous savons qu'une telle présentation devient vite lassante et monotone. De plus, le joueur, malgré la possibilité d'imaginer en pensée son propre décor, a du mal à entrer dans le jeu et ne fait que suivre l'action de l'extérieur.

Afin que le travail que vous aurez fourni pour entrer le listing complet de ce jeu (ou de celui que vous aurez créé) ne vous déçoive pas lors de son utilisation, il sera donc nécessaire d'y inclure des images. Le résultat sera alors surprenant de "professionnalisme" et d'intérêt.

Pour vous faciliter cette tâche, nous allons d'abord vous proposer un logiciel spécialisé dans la création d'images, puis, pour minimiser les ennuis de place mémoire, un compresseur/décompresseur d'images graphiques.

Dans un dernier temps, nous vous indiquerons le moyen d'inclure ces images dans le programme principal.

8. LE LOGICIEL CREIMAGE

Notre logiciel CREIMAGE d'aide à la création d'images graphiques va comporter les fonctions suivantes :

Choix de la palette de couleurs (couleurs associées aux encres 0,1,2,3).

Choix de la couleur de la bordure.

Choix de l'encre du curseur graphique.

Déplacement du curseur graphique, qui laissera sa trace derrière lui à l'aide des flèches du clavier.

Déplacement du curseur graphique sans laisser de trace en donnant les coordonnées absolues du point où l'on désire le placer.

Tracé d'une droite commençant au point où se trouve le curseur graphique et se terminant à un point dont on indiquera les coordonnées absolues, avec choix de l'encre de la droite.

Tracé d'une droite commençant au point où se trouve le curseur graphique et se terminant à un point dont on indiquera les coordonnées relatives par rapport à l'origine de la droite, avec choix de l'encre de la droite.

Tracé de figures : rectangle, triangle, cercle avec choix de la couleur.

Coloriage de zones : rectangle, triangle, cercle.

Copie d'une image à l'intérieur des mémoires de l'ordinateur, ce qui permettra de geler une image avant d'y effectuer des modifications lorsque l'on n'est pas sûr du résultat.

Restitution de l'image copiée précédente.

Écriture d'un fichier binaire de l'image sur disquette avec ou sans compression d'image (ou sur cassette).

Chargement d'un fichier binaire correspondant à une image compressée ou non.

Juxtaposition de deux images.

Création d'une image symétrique de l'image actuellement sur l'écran (image vue dans un miroir).

Permutation des couleurs associées à deux encres sans changement des couleurs de l'image.

Catalogue d'une disquette.

Effacement de n'importe quel fichier d'une disquette.

Aide-mémoire sur les couleurs disponibles et les couleurs actuellement choisies.

Aide-mémoire sur les fonctions disponibles de CREIMAGE.

Examinons tout d'abord la manière dont le programme et ses sous-programmes en langage machine vont être implantés.

☐ La zone d'adresses 49152 à 65535 correspond aux mémoires qui permettent de faire l'image à l'écran ; nous l'appellerons IMAGE1.

☐ La zone d'adresses 26204 à 42587 correspond aux mémoires qui permettent de copier l'image de l'écran ; nous l'appellerons IMAGE2.

☐ Pour exécuter les différentes fonctions décrites précédemment, il sera nécessaire de faire apparaître des messages à l'écran, messages qui se superposeront à l'image en cours de réalisation ; pour ne pas perdre cette image, nous la recopierons dans la zone d'adresses 9725 à 26108 que nous appellerons IMAGE3, puis les messages seront envoyés en superposition et, quand tous les messages seront terminés, nous recopierons IMAGE3 dans IMAGE1.

☐ Pour terminer, la zone d'adresses 367 à 9724 sera celle réservée au programme CREIMAGE et à ses variables associées. Pour cela, nous déclarerons un HIMEM de 9724. Mais la place mémoire limitée qui nous reste sera insuffisante pour exécuter des ordres SAVE ou LOAD ; en effet L'AMSDOS a besoin d'une zone mémoire libre située entre STREND et FRETOP (voir chapitre suivant pour l'explication de ces termes) dont la taille est environ de 8 K octets pour effectuer ses opérations d'écriture ou de lecture. Il est évident qu'avec 9 357 octets libres pour le programme nous ne pourrons pas disposer de cette place, nous ferons donc de l'allocation dynamique de mémoires, c'est-à-dire que nous modifierons le HIMEM en cours de travail du programme lorsque nous aurons à exécuter des ordres SAVE ou LOAD. Nous placerons dans ce cas le HIMEM en 26109, ce qui nous libérera les 16 K octets de IMAGE3. Quand les ordres SAVE ou LOAD auront été exécutés, nous replacerons le HIMEM en 9724. Cette opération n'est possible cependant qu'à condition de ne pas avoir besoin de variables alphanumériques pendant l'opération, car l'interpréteur BASIC se réfère à la valeur du HIMEM pour écrire ou lire les variables alphanumériques. Si on lui change cette valeur, il risque de ne pas retrouver ses variables ou de les écrire à un mauvais endroit.

Enfin la zone IMAGE2 sera aussi utilisée pour les opérations de

lecture ou d'écriture sur disquette ; deux cas différents suivant qu'il y aura ou non compression de l'image avant écriture ou décompression ou non après lecture (les compression/décompression d'image seront traitées plus loin dans ce livre) :

ÉCRITURE/LECTURE SANS COMPRESSION

Pour ne pas avoir le phénomène d'apparition des images par lignes entrelacées, nous commencerons par copier IMAGE1 dans IMAGE2, puis c'est cette IMAGE2 que nous écrirons sur la disquette sous forme d'un fichier binaire. Au chargement, ce fichier binaire va se placer dans la zone de IMAGE2, car un fichier binaire se charge à la place qu'il occupait au moment de l'écriture sur la disquette (en tête de fichier se trouve l'adresse d'origine et la longueur du fichier) et nous terminerons par une copie de IMAGE2 dans IMAGE1.

ÉCRITURE/LECTURE AVEC COMPRESSION

Le compresseur d'image que nous allons utiliser génère une image compressée à partir de l'adresse 26204, c'est-à-dire à la place de IMAGE2. La longueur de cette image compressée se trouvera dans les mémoires d'adresses 26189 et 26190 que nous pouvons lire avec des PEEK. Nous allons faire en sorte que l'image compressée ait une taille inférieure à 8 K octets, soit 8 192 octets pour accepter de faire l'écriture sur disquette. Dans ce cas, l'image compressée n'occupe que la moitié inférieure de la zone IMAGE2 ; nous appellerons cette zone IMAGE22, la seconde moitié supérieure sera appelée IMAGE21. Après vérification que la longueur de l'image compressée est inférieure à 8 K, nous copierons IMAGE22 dans IMAGE21 et c'est IMAGE21 que nous écrirons sur disquette. Cette manipulation est intéressante pour les programmes qui utiliseront les images compressées, il suffira de réserver uniquement la zone correspondant à IMAGE21 et d'effectuer la décompression en direction de IMAGE1, d'où un gain de place pour les programmes BASIC ou autres.

Par contre, la compression ne peut pas être effectuée à partir de l'adresse 34396 (début de IMAGE21), car on ne connaît pas a priori le résultat de la compression en longueur, celle-ci peut être supérieure à 8 K octets. Si l'on effectue la compression à partir de l'adresse 34396 et que la taille de l'image compressée est supérieure à 8 K octets, notre programme de compression va écrire dans la zone réservée

à l'interpréteur BASIC, ce qui aura pour effet de planter l'ordinateur et de vous faire "perdre la main". Vous ne pourrez retravailler qu'en arrêtant complètement l'ordinateur puis en le remettant sous tension.

Ce type de "plantage" peut bien sûr arriver aussi si l'image compressée a une taille supérieure à 16 K octets, ce qui heureusement est très rare. Pour être sûr du résultat, il faudrait effectuer la compression à partir du début de IMAGE3 pour avoir 32 K octets de place, mais cela poserait aussi des problèmes.

Avant d'étudier en détail le logiciel CREIMAGE, nous avons besoin de connaître un peu mieux notre Amstrad. D'autre part, nous serons aussi obligés de faire appel à des programmes en langage machine, nous donnerons donc quelques informations sur le fonctionnement du microprocesseur et des programmes en langage machine.

Pour finir, signalons qu'il existe quelques différences entre les Amstrad CPC 464, CPC 664 et CPC 6128, la seule qui aura une influence sur nos programmes provient d'un "BUG" dans l'interpréteur BASIC du CPC 464 (utilisation multiple de l'ordre MEMORY) ; ce BUG a été corrigé pour le CPC 6128, nous serons donc obligés de le "contourner" par une astuce dans le logiciel CREIMAGE pour CPC 464 (CREIMAGE utilise souvent l'ordre MEMORY). Le programme final de jeu d'aventure sera le même pour ces trois Amstrad, mais le logiciel CREIMAGE aura quelques différences.

9. QUELQUES RENSEIGNEMENTS SUR LE MICROPROCESSEUR DE L'AMSTRAD

■ LE MICROPROCESSEUR Z80

Un ordinateur est composé de différents éléments dont le point central est le microprocesseur. Dans le cas de l'Amstrad, il s'agit du Z80.

Le microprocesseur est capable d'effectuer certaines opérations logiques ou arithmétiques ainsi que d'échanger des informations avec les autres éléments qui constituent l'ordinateur. Nous parlons actuellement des possibilités données aux circuits électroniques qui le composent, et non des possibilités que l'on peut lui donner par des logiciels.

Voici le rôle de quelques-uns de ses composants, ces quelques notions nous seront utiles dans la suite de ce livre.

Dans un ordinateur, les types de mémoires suivants sont habituellement disponibles :

Les ROM, Read Only Memory

Mémoires accessibles uniquement en lecture, il n'est pas possible d'y écrire, sauf au moment de leur réalisation. Ce sont les mémoires mortes de l'ordinateur.

Les RAM, Random Access Memory

Mémoires accessibles en lecture et en écriture. Ce sont les mémoires vives de l'ordinateur.

Les registres

Mémoires de travail du microprocesseur, elles ont un temps d'accès plus court et certaines sont dotées d'un nom particulier.

L'ALU, Arithmetical Logical Unit

C'est le cœur du microprocesseur. Il est doté de la possibilité d'effectuer des opérations arithmétiques et logiques.

■ FONCTIONNEMENT DU Z80

Examinons maintenant les registres du Z80 :

L'accumulateur

Noté A, ce registre est associé à l'ALU pour lui permettre de travailler.

Les registres universels

Ils peuvent avoir plusieurs utilisations : ils ne sont pas affectés a priori à une tâche particulière.

Les registres d'adresse

Notés BC / DE / HL, ce sont des registres doubles : deux registres, indépendants physiquement, sont associés dans leur désignation pour la gestion des adresses ; le poids faible d'une adresse est contenu dans le premier registre, le poids fort dans le second. La notion de "poids faible" et de "poids fort" sera développée plus loin dans ce livre.

Le compteur ordinal

Noté PC, il contient l'adresse de la prochaine instruction à exécuter.

Le pointeur de pile

Noté SP, il permettra de gérer les piles.

Le registre d'index

Noté IX, il permettra de faire de l'adressage indexé.

Le registre d'indicateur

Ce registre possède des drapeaux positionnés par l'accumulateur en fonction du résultat de certaines opérations, par exemple le drapeau de résultat nul ou de débordement.

De plus, pour accélérer certains travaux, le Z80 possède deux jeux indépendants de certains des registres énumérés ci-dessus.

Revenons sur les registres d'adresses. Nous avons dit que ces registres étaient doubles ; cette particularité permet de gérer les adresses plus rapidement. Pour expliquer leur fonctionnement, commençons tout d'abord par quelques rappels sur les systèmes de numération.

NUMÉRATIONS DÉCIMALE, BINAIRE ET HEXADÉCIMALE

Lorsqu'on écrit le nombre 627 en numération décimale, la traduction arithmétique est :

$$6*10^2 + 2*10^1 + 7*10^0$$

10 est la base de la numération que l'on élève à la puissance 0, puis 1, puis 2, etc. (rappelons que 10 puissance 0 est égal à 1).

Le nombre 1011 en numération décimale se traduit par

$$1*10^3 + 0*10^2 + 1*10^1 + 1*2^0 = 1011$$

En numération binaire, il se traduit par

$$1*2^3 + 0*2^2 + 1*2^1 + 1*2^0$$

soit la valeur 14 en numération décimale.

En numération hexadécimale, il se traduit par

$$1*16^3 + 0*16^2 + 1*16^1 + 1*16^0$$

soit la valeur 1554 en numération décimale.

Un octet est un mot de huit bits, chaque bit peut prendre la valeur 0 ou 1 ; dans un octet, les nombres sont représentés en numération binaire. Le plus petit est 00000000 le plus grand est 11111111, ce qui nous donne en valeur décimale 0 pour le plus petit et 255 pour le plus grand.

$$1*2^7 + 1*2^6 + 1*2^5 + 1*2^4 + 1*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = 255$$

La numération hexadécimale est souvent utilisée lorsqu'on travaille directement sur le contenu d'un octet, car elle permet d'écrire tous les nombres possibles qu'il peut contenir avec seulement deux chiffres ou lettres.

La succession des chiffres en numération décimale est :

0,1,2,3,4,5,6,7,8,9

La succession des chiffres en numération hexadécimale est :

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

A a la valeur 10, B la valeur 11, C la valeur 12, D la valeur 13, E la valeur 14, F la valeur 15 en numération décimale.

Le nombre 255 en numération décimale s'écrit FF en numération hexadécimale : c'est le plus grand nombre que l'on peut écrire dans un octet.

$$F*16^1 + F*16^0 = 15*16 + 15 = 255$$

ÉCRITURE D'UNE ADRESSE

Le microprocesseur Z80 est capable de travailler avec des adresses allant de 0 à 65535. Nous venons de voir que dans un octet le nombre le plus grand que l'on pouvait représenter était 255 ; par conséquent, il sera nécessaire d'utiliser deux octets pour pouvoir représenter des nombres jusqu'à 65535. Ces deux octets sont utilisés de la manière suivante :

L'adresse sera égale à la valeur contenue dans le premier octet augmentée de la valeur contenue dans le second octet multipliée par 256.

Exemple

L'adresse 28712 sera représentée par les deux octets suivants :

00101000 01110000

L'octet 00101000 vaut 40 en décimal, l'octet 01110000 vaut 112 en décimal, on a bien en décimal 40 + 256*112 = 28712.

Le premier octet est dit contenir le poids faible, le second octet le poids fort.

■ LES MÉMOIRES DE L'AMSTRAD

Dans la version de base, l'Amstrad (464 ou 664) possède une mémoire RAM de 65 536 octets (soit 64K : 1K = 1 024 octets) ; mais toute cette mémoire n'est pas disponible pour l'utilisateur. Un certain nombre de mémoires est réservé pour le fonctionnement du microprocesseur, de l'interpréteur BASIC et les affichages à l'écran. Nous allons donner la carte des mémoires ainsi que quelques adresses remarquables utilisées par le microprocesseur ou l'interpréteur.

Pour les possesseurs d'un Amstrad 6128, la mémoire RAM est de 131 072 octets soit 128K, mais la mémoire est en réalité composée de deux blocs presque indépendants qui partagent les mêmes adresses. On appelle ces mémoires des mémoires *à bancs commutés.* A la mise sous tension ou après un reset complet par CTRL + SHIFT + ESC, on accède normalement au premier banc de mémoires. Pour accéder au second banc de mémoires il faut commuter des interrupteurs logiques *(soft switch)* qui sont constitués par des mémoires ; suivant les valeurs introduites dans ces mémoires, on accède au premier ou au second banc. Habituellement, le second banc de mémoires est utilisé comme un lecteur de disquette à accès rapide. Nos programmes n'utilisant pas

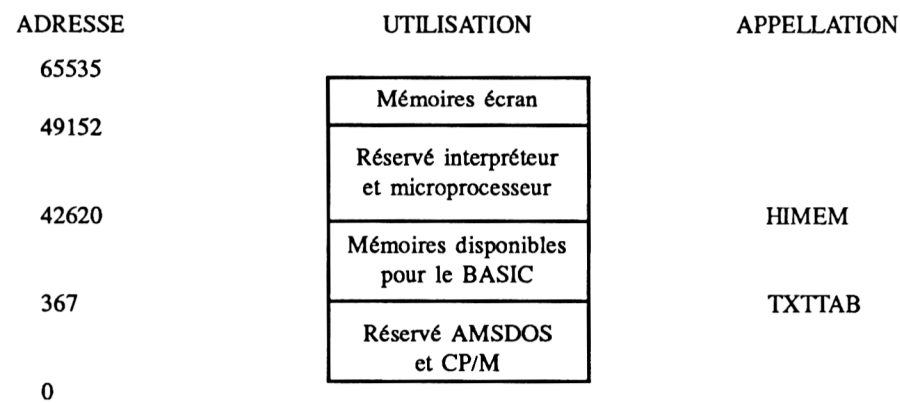
ces mémoires (pour être compatibles avec les différentes versions de l'Amstrad), nous ne donnerons pas d'autres renseignements sur leur fonctionnement et nous parlerons toujours du premier banc de mémoires.

CARTOGRAPHIE DES MÉMOIRES DE L'AMSTRAD

Les adresses des RAM vont de 0 à 65535.

- Les mémoires d'adresses 0 à 366 sont réservées pour le fonctionnement de l'AMSDOS et du CP/M.
- Les mémoires d'adresses 367 à 42619 sont libres pour le programme BASIC et ses variables associées.
- Les mémoires d'adresses 42620 à 49151 sont réservées au fonctionnement du microprocesseur et de l'interpréteur.
- Les mémoires d'adresses 49152 à 65535 sont réservées pour l'affichage écran.

La cartographie de la mémoire se schématise de la manière suivante :



L'adresse de début de la zone libre pour le programme BASIC s'appelle TXTTAB, cette adresse est stockée dans les mémoires réservées à l'interpréteur, soit 44673 (poids faible) et 44674 (poids fort) pour le CPC 464 et respectivement 44644 et 44645 pour les CPC 664 et 6128.

L'adresse de la fin de zone libre pour le programme BASIC s'appelle le HIMEM, cette adresse est stockée dans les mémoires réservées à

l'interpréteur soit 45199 (poids faible) et 45200 (poids fort) pour le CPC 464 et respectivement 45171 et 45172 pour les CPC 664 et 6128.

ADRESSES REMARQUABLES DE LA ZONE MÉMOIRE RÉSERVÉE A L'INTERPRÉTEUR

Les adresses remarquables que nous allons citer sont toutes implantées dans la zone réservée à l'interpréteur, nous les donnerons dans l'ordre poids faible-poids fort.

Nous venons de voir que l'adresse de début de la zone disponible pour le programme BASIC s'appelle TXTTAB et que l'adresse de la mémoire la plus haute (l'adresse la plus forte disponible) s'appelle le HIMEM, elle est stockée dans les mémoires d'adresses 45199 et 45200 pour le CPC 464 et respectivement 45171 et 45172 pour les CPC 664 et 6128. Elle peut être différente de la plus haute mémoire existante, caractéristique que nous utiliserons ultérieurement.

Remarque

Pour notre application, seule l'adresse de stockage de HIMEM nous sera utile ; pour plus de renseignements sur les autres adresses de stockage, nous vous conseillons de lire *Mise au point des programmes BASIC sur Amstrad* aux éditions Sybex.

L'adresse de fin de programme BASIC s'appelle le LOMEM.

Les variables numériques non indicées et l'adresse de stockage des variables alphanumériques non indicées sont stockées dans une zone de mémoires dont l'adresse de début est le LOMEM et l'adresse de fin le ARYTAB.

Les variables numériques indicées et les adresses de stockage des variables alphanumériques indicées sont stockées dans une zone dont l'adresse de début est ARYTAB et l'adresse de fin STREND.

Les variables alphanumériques, indicées ou non, sont stockées dans une zone dont l'adresse de début est HIMEM et dont l'adresse de fin est FRETOP. Il est à remarquer que pour cette zone l'adresse de début est supérieure à l'adresse de fin, à l'inverse des autres zones.

La zone allant de FRETOP à STREND est libre.

Il faut bien retenir de ce schéma que :

☐ Le LOMEM s'élève au fur et à mesure que le programme s'allonge.

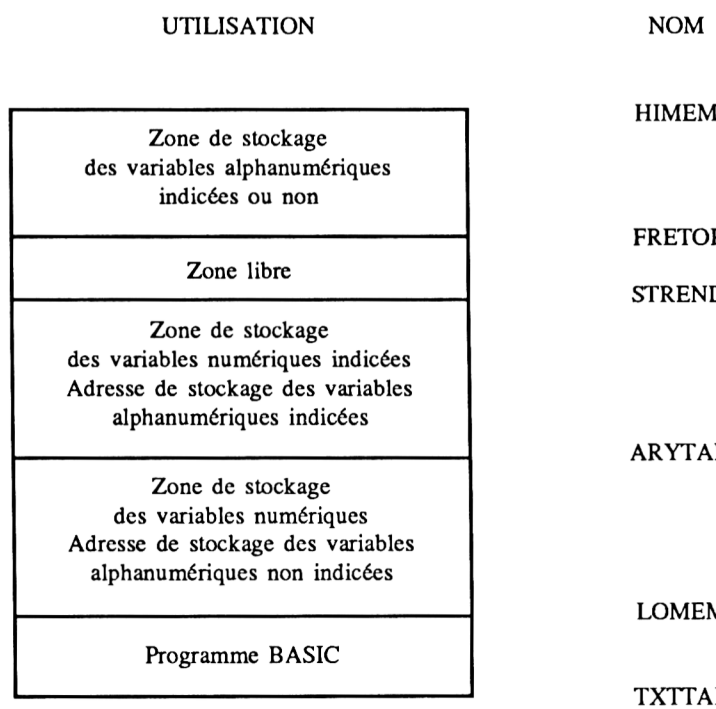
☐ La zone des variables qui s'étend de LOMEM à STREND est

d'autant plus importante que le nombre ou la dimension des variables sont importants.

☐ La zone supérieure qui va de HIMEM à FRETOP contient toutes les variables alphanumériques du programme (par variables il faut entendre tous les messages introduits par l'utilisateur, par exemple des INPUT R$, ou des concaténations telles que A$ = B$ + C$).

☐ Dès que la zone libre deviendra nulle, le programme n'aura plus de place pour stocker les variables, ce qui déclenchera par exemple le message d'erreur STRING SPACE FULL. Le programme sera donc arrêté dans son exécution. Il est donc nécessaire de limiter l'encombrement d'un programme et le nombre de variables utilisées, pour travailler correctement dans l'espace mémoire. Comme nous le verrons ultérieurement, nos logiciels seront directement limités par ces notions.

Pour la zone utilisable par le BASIC, nous avons la carte suivante :



10. L'AFFICHAGE GRAPHIQUE

Quelques rappels tout d'abord :

Un *pixel* est un point élémentaire de l'écran dont on peut choisir la couleur ; le nombre de pixels disponibles dépend des ordinateurs mais aussi du mode choisi dans le cas de l'Amstrad.

L'Amstrad possède trois modes graphiques :

- *Le mode 0,* dans lequel il y a 100 lignes ; chaque ligne possède 160 pixels. L'écran possède alors une définition de 100 x 160 = 16 000 pixels. D'autre part, chaque pixel peut avoir l'une des 16 couleurs définies par l'ordre BASIC INK ; ces 16 couleurs sont choisies parmi les 27 disponibles sur Amstrad.

- *Le mode 1,* dans lequel il y a 200 lignes ; chaque ligne possède 320 pixels. L'écran possède alors une définition de 200 x 320 = 64 000 pixels. D'autre part, chaque pixel peut avoir l'une des 4 couleurs définies par l'ordre BASIC INK ; ces 4 couleurs sont choisies parmi les 27 disponibles sur Amstrad.

- *Le mode 2,* dans lequel il y a 200 lignes ; chaque ligne possède 640 pixels. L'écran possède alors une définition de 200 x 640 = 128 000 pixels. D'autre part chaque pixel peut avoir l'une des 2 couleurs définies par l'ordre BASIC INK ; ces 2 couleurs sont choisies parmi les 27 disponibles sur Amstrad.

Lorsque nous disons 4 couleurs choisies parmi les 27 disponibles, cela veut dire qu'une image donnée possédera 4 couleurs, par exemple : jaune, noir, rouge et vert, mais une autre image pourra avoir 4 autres couleurs, par exemple : bleu, mauve, pourpre et blanc.

Pour nos applications d'images graphiques, le mode 0 est riche en couleurs, mais la finesse de l'image est insuffisante. Le mode 2 possède une excellente définition, mais il est pauvre en couleurs. Le mode 1 est un compromis entre les deux, c'est celui que nous choisissons pour notre application.

Représentation des images graphiques en mémoire

Pour faire apparaître les images sur l'écran du moniteur, l'Amstrad possède un microprocesseur particulier qui lit une partie des mémoires de l'Amstrad et en fonction, d'une part, du mode choisi et, d'autre part, des valeurs trouvées dans ces mémoires, allume les pixels avec les différentes couleurs choisies. Nous allons étudier maintenant la

correspondance qu'il y a entre les valeurs des mémoires qui représentent l'écran et la couleur des pixels.

Comme nous l'avons vu précédemment, les mémoires réservées à l'écran sont les mémoires qui vont de l'adresse 49152 à l'adresse 65535. Dans la suite de ce livre nous appellerons cette zone de mémoire : la *mémoire écran.* Nous allons explorer cette mémoire écran dans différents cas particuliers pour observer les valeurs prises par les octets en fonction de la couleur des pixels ; pour cela, introduisons le petit programme suivant :

```
10 INPUT "I= ", I
20 INPUT "C= ",C
30 INK I,C
40 FOR Y=0 TO 398 STEP 2
50 MOVE 0,Y:DRAW 639,Y,I
60 NEXT
70 A1$=BIN$(PEEK(49152),8)
80 A2$=BIN$(PEEK(49153),8)
90 A3$=BIN$(PEEK(49154),8)
100 A4$=BIN$(PEEK(49155),8)
110 A5$=BIN$(PEEK(49156),8)
120 A6$=BIN$(PEEK(49157),8)
130 A7$=BIN$(PEEK(49158),8)
140 A8$=BIN$(PEEK(49159),8)
150 A9$=BIN$(PEEK(49160),8)
160 PRINT A1$,A2$,A3$
170 PRINT A4$,A5$,A6$
180 PRINT A7$,A8$,A9$
```

Exécutons ce programme et donnons tout d'abord la valeur 0 à notre variable I et 1 à notre variable C ; notre écran va donc se colorer avec la couleur d'encre 0 à laquelle nous venons d'associer le bleu, puis nous allons observer les valeurs en binaire des 9 premiers octets de la mémoire écran soit :

00000000	00000000	00000000
00000000	00000000	00000000
00000000	00000000	00000000

Tous les octets ont la même valeur et cette valeur est 00000000.

Donnons maintenant comme valeurs I = 0 et C = 3, l'écran se colore en rouge, et nous obtenons comme valeurs :

00000000	00000000	00000000
00000000	00000000	00000000
00000000	00000000	00000000

Les octets de la mémoire écran n'ont pas changé de valeur, ils dépendent donc uniquement de l'encre choisie et non de sa couleur.

Donnons maintenant comme valeurs I = 1 et C = 3, l'écran se colore en rouge, et nous obtenons comme valeurs :

11110000	11110000	11110000
11110000	11110000	11110000
11110000	11110000	11110000

Donnons maintenant comme valeurs I = 2 et C = 3, l'écran se colore en rouge, et nous obtenons comme valeurs :

00001111	00001111	00001111
00001111	00001111	00001111
00001111	00001111	00001111

Donnons maintenant comme valeurs I = 3 et C = 3, l'écran se colore en rouge, et nous obtenons comme valeurs :

11111111	11111111	11111111
11111111	11111111	11111111
11111111	11111111	11111111

Modifions maintenant notre programme de la manière suivante :

```
10 CLS
20 PLOT 0,398,1:PLOT 2,398,1
30 PLOT 4,398,1:PLOT 6,398,1
70 A1$=BIN$(PEEK(49152),8)
80 A2$=BIN$(PEEK(49153),8)
90 A3$=BIN$(PEEK(49154),8)
100 A4$=BIN$(PEEK(49155),8)
```

```
110 A5$=BIN$(PEEK(49156),8)
120 A6$=BIN$(PEEK(49157),8)
130 A7$=BIN$(PEEK(49158),8)
140 A8$=BIN$(PEEK(49159),8)
150 A9$=BIN$(PEEK(49160),8)
160 PRINT A1$,A2$,A3$
170 PRINT A4$,A5$,A6$
180 PRINT A7$,A8$,A9$
```

Exécutons alors ce nouveau programme qui doit allumer quatre points avec la couleur 1 et nous donner toujours la valeur des 9 premiers octets :

11110000	00000000	00000000
00000000	00000000	00000000
00000000	00000000	00000000

Nous pouvons constater que seul le premier octet a changé de valeur par rapport à un écran vide ; en effet, en mode 1 un octet représente 4 pixels. Examinons le procédé de codage :

Un octet comporte 8 bits qui peuvent prendre les valeurs 0 ou 1 (après une instruction BIN$ le résultat se lit de la droite vers la gauche), nous numérotons les bits d'un octet de la droite vers la gauche soit : B0, B1, B2, B3, B4, B5, B6 et B7, le bit le plus à droite est B0, le bit le plus à gauche est B7. En mode 1, la valeur de l'encre choisie pour un pixel est donnée par la valeur des couples (B0,B4), (B1,B5), (B2,B6) et (B3,B7). Les quatre premiers pixels en haut et à gauche de l'écran sont représentés par la mémoire 49152, le premier est représenté par le couple de bits (B0,B4) de l'octet contenu dans cette mémoire, le deuxième par le couple (B1,B5), le troisième par le couple (B2,B6) et le quatrième par le couple (B3,B7). Chaque couple peut prendre l'une des quatre valeurs suivantes : (0,0), (0,1), (1,0) et (1,1). On a donc quatre possibilités correspondant aux quatre valeurs possibles pour l'encre ; l'encre 0 correspondra au couple (0,0), l'encre 1 au couple (0,1), l'encre 2 au couple (1,0) et l'encre 3 au couple (1,1). Notre première mémoire affichée par notre programme comporte bien 4 couples (0,1). Prenons un autre exemple :

Si le premier pixel de cette série de quatre est coloré avec l'encre n° 0, le bit B0 = 0 et le bit B4 = 0 ; si le deuxième est coloré avec l'encre n° 1, le bit B1 = 0 et le bit B5 = 1 ; si le troisième est coloré avec l'encre 2 le bit B2 = 1 et le bit B6 = 0 et pour finir si le quatrième pixel est coloré avec l'encre 3 le bit B3 = 1 et le bit B7 = 1, on obtient

alors pour l'octet entier la valeur binaire : 00110101, soit la valeur décimale 172. Vérifions cela à l'aide du programme suivant :

```
10 CLS
20 PLOT 0,398,0:PLOT 2,398,1
30 PLOT 4,398,2:PLOT 6,398,3
70 A1$=BIN$(PEEK(49152),8)
80 A2$=BIN$(PEEK(49153),8)
90 A3$=BIN$(PEEK(49154),8)
100 A4$=BIN$(PEEK(49155),8)
110 A5$=BIN$(PEEK(49156),8)
120 A6$=BIN$(PEEK(49157),8)
130 A7$=BIN$(PEEK(49158),8)
140 A8$=BIN$(PEEK(49159),8)
150 A9$=BIN$(PEEK(49160),8)
160 PRINT A1$,A2$,A3$
170 PRINT A4$,A5$,A6$
180 PRINT A7$,A8$,A9$
```

On obtient bien la valeur 10101100.

Examinons maintenant l'ordre de stockage des pixels dans la mémoire écran. Nous numéroterons les lignes de 0 à 199 du haut vers le bas, et les pixels sur les lignes seront numérotés de 0 à 319 (à noter que les coordonnées graphiques de l'Amstrad donnent X = 0 et Y = 0 pour le point en bas à gauche, X = 638 Y = 0 pour le point en bas à droite, X = 0 Y = 398 pour le point en haut à gauche et X = 638 Y = 398 pour le point en haut à droite).

Dans la mémoire d'adresses 49152 à 49231, nous trouvons les 80 octets de la ligne n° 0, puis de 49232 à 49311 les octets de la ligne n° 8, puis de 49312 à 49391 les octets de la ligne n° 16, et ainsi de suite jusqu'à la ligne n° 192 qui se termine en 51151. Les octets d'adresses 51152 à 51199 ne sont pas utilisés. A partir de la mémoire d'adresse 51200, nous trouvons les octets de la ligne n° 1, puis n° 2, et ainsi de suite jusqu'à la ligne n° 193, puis *n* octets inutilisés, puis la ligne n° 3, etc. La mémoire écran contient donc les lignes par paquets de 25, et dans un paquet de 25 les lignes ne sont pas des lignes successives mais des lignes espacées de huit en huit. Entre chaque paquet de 25, il y a 48 mémoires inutilisées.

Nous allons examiner ce schéma de lignes entrelacées à l'aide du programme suivant :

```
10 FOR I=49152 TO 65535
20 POKE I,15
30 NEXT
```

La manière dont l'écran se colore met bien en évidence l'entrelacement des lignes.

Dans le cas du mode 2 nous avons deux fois plus de pixels à contrôler à l'écran, mais le nombre de mémoires disponibles reste identique, par conséquent un octet représentera 8 pixels et donc un bit dans un octet représentera 1 pixel. Un bit pouvant prendre les valeurs 0 ou 1, le pixel n'aura que deux encres possibles.

Dans le cas du mode 0, nous avons au contraire quatre fois moins de pixels que dans le mode 1 ; un octet représentera un pixel, et nous aurons, en théorie du moins, 256 encres disponibles. L'Amstrad est cependant limité à 27 encres disponibles (si le mode 0 avait eu 32 000 pixels, un octet aurait représenté 2 pixels qui auraient eu 16 encres possibles).

11. LA RECOPIE DE MÉMOIRES

Pour les besoins de notre logiciel d'aide à la création d'images, nous avons besoin d'effectuer des recopies d'images à l'intérieur des mémoires de l'Amstrad. Nous allons tout d'abord écrire un programme en BASIC pour recopier la zone mémoire écran qui va de l'adresse 49152 à l'adresse 65535, dans la zone mémoire qui va de l'adresse 26204 à l'adresse 42587. Voici ce programme :

```
10 MEMORY 26000:MODE 1:CLS
20 MOVE 10,0:DRAW 10,400,1:MOVE 20,0:DRA
W 20,400,2:MOVE 28,0:DRAW 28,400,3:LOCAT
E 15,10:PRINT "ESSAI"
30 FOR I=49152 TO 65534:POKE I-22948,PEE
K(I):LOCATE 10,24:PRINT I:NEXT
40 MODE 1
50 FOR I=26204 TO 42587:POKE I+22948,PEE
K(I):NEXT:PRINT "FIN"
```

Ligne 10

Nous changeons la place du HIMEM, passons en mode 1 et nettoyons l'écran.

Ligne 20

Ces quelques ordres graphiques permettent de tracer une image test sur l'écran.

Ligne 30

La boucle en I recopie le contenu des mémoires dont l'adresse va de 49152 à 65535, dans les mémoires dont l'adresse va de 26204 à 42587. Comme cette opération est assez longue en BASIC, nous avons ajouté un ordre d'affichage à l'écran, qui donne le numéro de la mémoire en cours de transfert (ce qui ralentit notablement le programme, mais permet de vérifier que le travail est en cours).

Ligne 40

Nettoyage de l'écran, et par conséquent remise à zéro des mémoires 49152 à 65535.

Ligne 50

Nous effectuons le travail en sens inverse, en recopiant les mémoires dont l'adresse est comprise entre 26204 et 42587 dans les mémoires qui vont de 49152 à 65535. Nous ne mettons pas de PRINT de l'adresse

de la mémoire en cours de recopie, car le dessin qui réapparaît à l'écran nous permet de suivre le travail. Vous avez constaté au passage que la recopie est très lente (environ 10 minutes), ce qui est prohibitif pour notre application. Nous allons donc recourir au langage machine (ou à l'assembleur pour ceux qui possèdent un assembleur/désassembleur).

Tout d'abord, commençons par donner quelques explications sur ce qu'est le langage machine. Les microprocesseurs sont capables d'exécuter directement, sans l'aide de logiciels, un certain nombre d'ordres élémentaires. La liste de ces ordres et leur fonction se trouvent dans la documentation des microprocesseurs ou dans des livres traitant de leur programmation (par exemple, pour le Z80 : *Programmation du Z80* par Rodnay Zaks aux éditions Sybex), ou bien dans des livres traitant d'un assembleur pour votre ordinateur.

Un assembleur est un logiciel qui vous aide à programmer en langage machine. Pour de petits programmes, on peut se passer d'assembleur. A chaque ordre exécutable par le microprocesseur correspondent, pour le Z80, une ou plusieurs valeurs à introduire dans les mémoires de votre ordinateur. Après introduction à l'aide d'ordres POKE des ordres correspondant à votre programme, il suffit de faire effectuer un saut à la première mémoire qui contient le programme avec l'ordre BASIC CALL pour que votre programme s'exécute. On affecte à chaque ordre en langage machine un mnémonique permettant de mieux comprendre le programme lorsqu'on l'écrit. Un logiciel d'assembleur permet de taper directement les mnémoniques pour introduire un programme dans les mémoires, au lieu de faire l'introduction des valeurs correspondantes avec des POKE. Mais plutôt que de continuer à donner des explications théoriques, réalisons notre programme de transfert d'image.

Le Z80 possède un ordre très puissant qui va effectuer la recopie précédente en moins d'une seconde ; cet ordre a comme mnémonique LDIR. L'ordre LDIR effectue le travail suivant : il recopie le contenu de la mémoire dont l'adresse se trouve dans le registre double HL dans la mémoire dont l'adresse se trouve dans le registre DE. Puis il ajoute 1 au contenu du registre DE et du registre HL et il retranche 1 au contenu du registre BC. Il recommence ensuite le même travail, et ce jusqu'à ce que le registre BC ait la valeur 0. Pour effectuer le travail, cet ordre a donc besoin de connaître l'adresse de la première mémoire à transférer, l'adresse de la première mémoire de destination et, bien sûr, le nombre de mémoires à recopier.

L'adresse de la première mémoire à recopier doit donc être introduite

dans le registre double HL. Pour effectuer cela, nous utiliserons l'ordre LD HL, qui veut dire "charger le registre double HL avec les valeurs contenues dans les deux octets qui suivent" (dans l'ordre poids faible, poids fort). Notre première adresse est celle de l'IMAGE 1, soit 49152, ce qui s'écrit LD HL,49152. L'ordre LD HL a comme valeur 33. On obtient alors pour l'ordre "Charger le registre double HL avec la valeur 49152":

En assembleur LD HL,49152

En langage machine 33-0-192 (0 + 256*192 = 49152)

L'adresse de la première mémoire de destination doit être introduite dans le registre double DE. Pour effectuer cela, nous utiliserons l'ordre LD DE, qui veut dire "charger le registre double DE avec les valeurs contenues dans les octets qui suivent" (dans l'ordre poids faible, poids fort). Notre première adresse est 26204, cela s'écrit LD DE,26204. L'ordre LD DE a comme valeur 17. On obtient alors pour l'ordre "Charger le registre double DE avec la valeur 26204" :

En assembleur LD DE,26204

En langage machine 17-92-102 (92 + 256*102 = 26204)

Le nombre de mémoires à recopier doit être introduit dans le registre double BC. Pour effectuer cela, nous utiliserons l'ordre LD BC, qui veut dire "charger le registre double BC avec les valeurs contenues dans les octets qui suivent" (dans l'ordre poids faible, poids fort). Une image correspond à 16 384 octets, ce qui s'écrit LD BC,16384. L'ordre LD BC a comme valeur 1. On obtient alors pour l'ordre "Charger le registre double BC avec la valeur 16384" :

En assembleur LD BC, 16384

En langage machine 1-0-32 (0 + 256*64 = 16384)

Lorsque ce programme sera terminé, il faudra ensuite revenir au programme BASIC, ce qui s'obtiendra par l'ordre de mnémonique RET et de valeur 201.

Nous allons implanter ce programme à partir de la mémoire 26192 à l'aide du programme BASIC suivant qui donne en REM la traduction en assembleur du langage machine introduit :

```
10 MEMORY 26000:MODE 1
20 MOVE 10,0:DRAW 10,400,1:MOVE 20,0:DRA
W 20,400,2:MOVE 28,0:DRAW 28,400,3:LOCAT
E 15,10:PRINT "ESSAI"
100 POKE 26192,1
110 POKE 26193,0
120 POKE 26194,64
130                          REM LD BC,16384
140 POKE 26195,17
150 POKE 26196,92
160 POKE 26197,102
170                          REM LD DE,26204
180 POKE 26198,33
190 POKE 26199,0
200 POKE 26200,192
210                          REM LD HL,49152
220 POKE 26201,237
230 POKE 26202,176
240                          REM LDIR
250 POKE 26203,201
260                          REM RET
270 CALL 26192
280 MODE 1
300 POKE 26192,1
310 POKE 26193,0
320 POKE 26194,64
330                          REM LD BC,16384
340 POKE 26195,17
350 POKE 26196,0
360 POKE 26197,192
370                          REM LD DE,26204
380 POKE 26198,33
390 POKE 26199,92
400 POKE 26200,102
410                          REM LD HL,49152
420 POKE 26201,237
430 POKE 26202,176
440                          REM LDIR
```

```
450 POKE 26203,201
460                          REM RET
470 CALL 26192
```

Nous avons, en tête de ce programme, les mêmes instructions (10 et 20) que pour le premier programme, puis le programme en langage machine (100 à 260) pour effectuer le transfert des mémoires dont les adresses vont de 49152 à 65535 vers les mémoires dont les adresses vont de 26204 à 42587, appel de ce programme à l'aide de l'ordre CALL (270), nettoyage de l'écran (280), de nouveau le programme en langage machine, mais cette fois pour effectuer le transfert des mémoires dont les adresses vont de 26204 à 42587 vers les mémoires dont les adresses vont de 49152 à 65535 (300 à 460), et appel de nouveau du programme (470).

Lorsque vous exécutez ce programme, vous avez apparition de l'image, transfert, nettoyage écran, rappel de l'image par transfert. L'ensemble de ces opérations est presque instantané.

12. LA COMPRESSION/ DÉCOMPRESSION D'IMAGE

Nous allons maintenant aborder le problème de stockage des images sur disquette (ou sur cassette).

La procédure de stockage que nous avons utilisée précédemment donne naissance sur la disquette à un fichier d'une taille de 17K octets. Notre jeu d'aventure va comporter 66 images différentes. Chaque face de disquette peut contenir 170K octets, donc 10 images. Nous avons donc besoin de 7 faces de disquette. Nous allons essayer d'améliorer la situation en effectuant une compression/décompression d'image.

Pour effectuer une compression d'image, nous allons coder l'image d'une manière différente. Nous allons explorer la mémoire écran pour identifier les différents octets qui composent l'image, et lorsque nous rencontrerons N octets successifs possédant la même valeur, au lieu de garder N fois cette valeur nous indiquerons le nombre N suivi de la valeur de l'octet. Comme nous ne pourrons pas faire la différence entre les valeurs de N et les valeurs d'un octet de la mémoire écran, nous serons obligés aussi de coder les octets uniques par la valeur 1 suivie de la valeur de l'octet. Au fur et à mesure de l'exploration de la mémoire écran, nous écrirons le résultat de notre codage dans les mémoires de l'Amstrad à un endroit différent de la mémoire écran.

L'ensemble des octets obtenus composera notre nouvelle image et c'est ce résultat que nous inscrirons sur la disquette. En sens inverse, lorsque nous désirerons faire apparaître une image à l'écran, nous lirons le fichier "image compressée" sur la disquette, puis nous traduirons en sens inverse les valeurs introduites dans les mémoires pour les écrire sous forme traditionnelle dans la mémoire écran et voir apparaître l'image.

S'il n'existe pas dans toute la mémoire écran deux octets successifs identiques, le résultat de notre codage consistera à doubler la taille du fichier, car nous aurons traduit chaque octet par deux octets. Heureusement, dans le cas général, une image possède des zones d'une seule couleur ou avec un motif répétitif (exemple rouge, bleu, rouge, bleu...), ce qui correspond à des lignes écran composées d'octets successifs identiques.

Dans un octet, la valeur maximale que l'on peut écrire est 255, par conséquent si nous voulons coder le nombre d'octets successifs identiques rencontrés dans l'exploration nous devrons limiter N à 255 et si N est plus grand, nous le coderons en deux fois. Par exemple, si 520 octets successifs sont identiques et de valeur 12, le résultat du codage sera 255-12-255-12-10-12 ; dans ce cas simple, la compression fait passer de 520 octets à 6 octets.

Nous allons tout d'abord réaliser un compresseur graphique en BASIC pour bien comprendre son fonctionnement. Sa vitesse d'exécu-

tion sera très lente et nous réaliserons ensuite un programme en langage machine qui sera d'une rapidité étonnante par rapport au programme en BASIC.

Nous effectuerons le travail de compression en écrivant le résultat du codage à partir de la mémoire d'adresse 26204.

Voici le listing du programme en langage BASIC.

```
10 MEMORY 26000:MODE 1:CLS
20 J=1:K=26204:L=0:MOVE 10,0:DRAW 10,400
,1:MOVE 20,0:DRAW 20,400,2:MOVE 28,0:DRA
W 28,400,3:LOCATE 15,10:PRINT "ESSAI"
30 FOR I=49152 TO 65534:A=PEEK(I):B=PEEK
(I+1)
40 IF J=1 THEN C=A
50 IF A=B THEN J=J+1
60 IF A<>B OR J=255 THEN POKE K,J:POKE K
+1,C:K=K+2:J=1:L=L+2
70 LOCATE 10,24:PRINT I:NEXT I
80 POKE K,J:POKE K+1,C
90 SAVE "IM",B,26204,L
100 CLS:PRINT L:INPUT R
110 MODE 1:CLS
120 LOAD "IM"
130 A=49151:B=26204
140 FOR K=1 TO PEEK(B):POKE A+K,PEEK(B+1
):NEXT K
150 A=A+K-1:IF A<65533 THEN B=B+2:GOTO 1
40
```

Lignes 10 et 20

Les ordres habituels pour positionner le HIMEM et réaliser un petit dessin graphique. De plus, nous initialisons la variable J à la valeur 1 ; J nous permettra de compter combien d'octets successifs sont identiques. Nous posons K = 26203.

Ligne 30

Début d'une boucle en I pour explorer toute la mémoire écran ; A et B contiennent la valeur de deux octets successifs.

Ligne 40

Nous mémorisons dans la variable C la valeur du premier octet rencontré au début d'une série, ce qui correspond à la valeur 1 de J.

Ligne 50

Si deux octets successifs introduits dans A et B sont égaux, nous incrémentons J d'une unité.

Ligne 60

Si les deux octets successifs sont différents, ou si la série rencontrée comporte 255 octets égaux, nous inscrivons dans la zone mémoire image compressée le nombre d'octets successifs égaux et la valeur commune de ces octets. Puis nous incrémentons la variable K de deux unités, nous repositionnons la variable J à 1 pour le décompte de la nouvelle série.

Ligne 70

Pour faire passer le temps, nous affichons l'adresse de la mémoire en cours d'exploration. Le NEXT de fin de boucle.

Ligne 80

Lorsque nous arrivons à la dernière mémoire de la zone écran, dans le cas général la boucle précédente n'inscrit pas ce dernier résultat, nous l'inscrivons donc ici...

Ligne 90

... et nous envoyons le résultat sur disquette sous forme d'un fichier binaire de nom IM.

Lignes 100 à 120

Nettoyage écran ; on imprime le résultat de la compression et l'ordre INPUT permet d'arrêter le programme le temps de lire cette information. Nettoyage des mémoires écran, de l'écran, et chargement de l'image compressée.

Ligne 130

A est initialisé à la valeur de la première adresse de la zone écran, moins une unité, et B à la valeur de la première adresse de l'image compressée.

Ligne 140

Boucle en K pour décompresser l'image, PEEK(B) contient le nombre d'octets successifs égaux et PEEK(B + 1) la valeur de ces octets ; on va donc inscrire cette valeur PEEK(B) fois dans la zone écran.

Ligne 150

On passe au couple "nombre d'octets, valeur de l'octet suivant" et on recommence jusqu'à ce que l'image écran soit complète, c'est-à-dire jusqu'à ce qu'on ait introduit des valeurs dans la zone écran complète, ce qui est obtenu lorsque A = 65535.

Vous exécutez ce programme : le temps de compression est supérieur à 10 minutes, ce qui est prohibitif pour notre application.

Passons maintenant au compresseur en langage machine, introduit à l'aide d'un programme BASIC qui donne des renseignements sur le programme introduit. Voici ce programme :

```
10 REM COMPRESSEUR/DECOMPRESSEUR
20 REM EN ASSEMBLEUR
30 MEMORY 26130:MODE 1
40 MOVE 0,0
50 DRAW 639,0,1:DRAW 639,399,1
60 DRAW 0,399,1:DRAW 0,0,1
70 MOVE 40,0:DRAW 40,400,1
80 MOVE 20,0:DRAW 20,400,2
90 MOVE 28,0:DRAW 28,400,3
100 LOCATE 15,10:PRINT "ESSAI"
110 REM
120 REM COMPRESSEUR
130 REM
140 FOR I=26132 TO 26187
150 READ A:POKE I,A:NEXT I
160 DATA 33,0,192
170                 REM 26132 LD HL,49152
180 DATA 1,91,102
190                 REM 26135 LD BC,26203
200 DATA 22,1
210                 REM 26138 LD D,1
220 DATA 94
230                 REM 26140 LD E,(HL)
240 DATA 35
250                 REM 26141 INC HL
260 DATA 125
270                 REM 26142 LD A,L
280 DATA 238,255
290                 REM 26143 XOR,255
300 DATA 194,42,102
```

```
310                 REM 26145 JP NZ,26154
320 DATA 124
330                 REM 26148 LD A,H
340 DATA 238,255
350                 REM 26149 XOR,255
360 DATA 202,66,102
370                 REM 26151 JP Z,26178
380 DATA 123
390                 REM 26154 LD A,E
400 DATA 190
410                 REM 26155 CP (HL)
420 DATA 194,57,102
430                 REM 26156 JP NZ,26169
440 DATA 20
450                 REM 26159 INC D
460 DATA 202,54,102
470                 REM 26160 JP Z,26166
480 DATA 195,29,102
490                 REM 26163 JP,26141
500 DATA 22,255
510                 REM 26166 LD D,255
520 DATA 0
530                 REM 26168 NOP
540 DATA 03
550                 REM 26169 INC BC
560 DATA 123
570                 REM 26170 LD A,E
580 DATA 02
590                 REM 26171 LD (BC),A
600 DATA 122
610                 REM 26172 LD A,D
620 DATA 03
630                 REM 26173 INC BC
640 DATA 02
650                 REM 26174 LD (BC),A
660 DATA 195,26,102
670                 REM 26175 JP 36816
680 DATA 03
690                 REM 26178 INC BC
700 DATA 02
710                 REM 26179 LD (BC),A
720 DATA 122
```

```
730                 REM 26180 LD A,D
740 DATA 03
750                 REM 26181 INC BC
760 DATA 02
770                 REM 26182 LD (BC),A
780 DATA 237,67,77,102
790                 REM 26183 LD 26189,BC
800 DATA 201
810                 REM 26187 RET
820 REM
830 REM DECOMPRESSEUR
840 REM
850 FOR I=42589 TO 42618
860 READ A:POKE I,A:NEXT I
870 DATA 33,0,192
880                 REM 42589 LD HL,49152
890 DATA 1,91,134
900                 REM 42592 LD BC,34395
910 DATA 3
920                 REM 42595 INC BC
930 DATA 10
940                 REM 42596 LD A,(BC)
950 DATA 87
960                 REM 42597 LD D,A
970 DATA 3
980                 REM 42598 INC BC
990 DATA 10
1000                REM 42599 LD A,(BC)
1010 DATA 114
1020                REM 42600 LD (HL),D
1030 DATA 35
1040                REM 42601 INC HL
1050 DATA 61
1060                REM 42602 DEC A
1070 DATA 194,104,166
1080                REM 42603 JP NZ,42600
1090 DATA 125
1100                REM 42606 LD A,L
1110 DATA 238,255
1120                REM 42607 XOR,255
1130 DATA 194,99,166
1140                REM 42609 JP NZ,42595
```

```
1150 DATA 124
1160                REM 42612 LD A,H
1170 DATA 238,255
1180                REM 42613 XOR,255
1190 DATA 194,99,166
1200                REM 42615 JP NZ,42595
1210 DATA 201
1220                REM 42618 RET
1230 CALL 26132
1240 POKE 26192,1:POKE 26193,0:POKE 2619
4,32:POKE 26195,17:POKE 26196,92:POKE 26
197,134:POKE 26198,33:POKE 26199,92:POKE
 26200,102:POKE 26201,237:POKE 26202,176
:POKE 26203,201:CALL 26192
1250 NB=PEEK(26189)+256*PEEK(26190)-2620
3
1260 SAVE "IMP",B,34396,NB
1270 PRINT "FIN"
1280 MODE 1:LOAD "IMP":CALL 42589
1290 FOR I=1 TO 4000:NEXT I
1300 END
```

Lignes 10 à 100

Positionnement du HIMEM, nettoyage écran et réalisation d'un dessin de test.

Lignes 140 et 150

Boucle en I pour charger les valeurs de notre programme de compression en langage machine.

Les lignes suivantes se lisent par deux : la première correspond aux valeurs en langage machine, la seconde à la traduction sous forme de mnémoniques assembleurs avec en tête l'adresse de la mémoire où est implanté l'ordre.

PROGRAMME DE COMPRESSION

26132 LD HL,49152

Charge le registre double HL avec la valeur 49152, première adresse de la zone image écran que l'on va compresser. Le registre double HL donnera l'adresse du prochain octet à comparer.

26135 LD BC,26203

Charge le registre double BC avec la valeur 26203, première adresse, moins une unité, de la zone où sera inscrit le résultat de la compression. Le registre double BC donnera l'adresse où écrire le résultat de la compression d'une série d'octets identiques.

26138 LD D,1

Charge le registre simple D avec la valeur 1. Le registre D sera le compteur d'octets identiques successifs.

26140 LD E,(HL)

Charge le contenu de la mémoire dont l'adresse est inscrite dans le registre double HL, dans le registre E ; c'est la valeur du premier octet d'une éventuelle série.

26141 INC HL

Incrémente le contenu du registre double HL d'une unité.

Nous allons effectuer maintenant le test qui permettra de savoir si nous avons terminé la compression. Cela correspond au fait que le registre double HL contient la valeur 65535. En langage machine, on ne peut pas sur le Z80 effectuer un test de valeur directement sur un registre double. Il faut donc effectuer le test en deux fois : si le registre double HL contient la valeur 65535, chaque registre H et L contient la valeur 255 (255 + 256*255 = 65535). D'autre part, la comparaison ne peut être effectuée qu'avec une valeur contenue dans l'accumulateur A. Nous allons recopier L dans A, le comparer avec 255 et, suivant le résultat, nous continuerons le traitement ou nous irons vérifier si le contenu de H est ou non égal à 255 en le transférant à son tour dans A. Si le contenu de H est différent de 255, on continue l'exploration ; sinon la compression est terminée.

26142 LD A,L

Charge l'accumulateur A avec le contenu du registre L.

26143 XOR,255

Effectue un OU exclusif du contenu de l'accumulateur A avec la valeur 255. Le résultat de l'opération positionne des indicateurs, notamment celui de "résultat égal à 0", ce qui sera obtenu si A contient la valeur 255.

26145 JP NZ,26154

Saut conditionnel à l'adresse 26154. La condition utilisée est "Non Zéro". En conséquence, si le contenu de A est égal à 255, le résultat est Zéro et le saut ne s'effectue pas ; par contre, si le contenu est différent de 255, le résultat est Non Zéro et le saut s'effectue. Le saut alors effectué permet de ne pas faire le test sur le contenu de H.

26148 LD A,H

Charge H dans A.

26149 XOR,255

OU exclusif avec 255.

26151 JP Z,26178

Cette fois-ci, nous faisons un saut conditionnel sur la valeur Zéro ; en effet, si on a Zéro c'est que H est égal à 255, et comme L est déjà égal à 255, le travail de compression est terminé et nous allons en 26178 pour sortir du programme.

26154 LD A,E

Dans E nous avons la valeur de la mémoire dont l'adresse était contenue dans HL avant son incrémentation, c'est-à-dire le premier octet d'une éventuelle série de la mémoire écran ; nous copions cette valeur dans A.

26155 CP (HL)

Dans HL nous avons l'adresse de l'octet suivant de la mémoire écran. Nous faisons la comparaison de la valeur de cet octet avec celui contenu dans A (premier octet de la série). Si les deux octets sont égaux, l'indicateur de Zéro est positionné à Zéro ; dans le cas contraire, il est positionné à Non Zéro.

26156 JP NZ,26169

Saut conditionnel, si Non Zéro, on va en 26169 pour inscrire le résultat de cette exploration ; si Zéro, les deux octets sont égaux et nous allons incrémenter le compteur qui est D d'une unité à l'ordre suivant.

26159 INC D

Incrémentation de D d'une unité.

26160 JP Z,26166

Si, au cours de l'incrémentation précédente de D, nous dépassons 255, l'indicateur passe à Zéro, cela indique que le maximum d'octets successifs égaux que nous pouvons prendre en compte a été dépassé. Nous faisons donc un saut à l'adresse qui permet d'inscrire le résultat. Le dernier octet comparé deviendra ensuite l'octet de référence de la prochaine suite. Au passage, nous remettrons D à la valeur 255 puisque ce dernier octet n'appartiendra pas à la suite que nous allons inscrire.

26163 JP,26141

Saut inconditionnel à l'adresse 26141, c'est la fin de la boucle et nous retournons tester l'octet suivant de la mémoire écran.

26166 LD D,255

Comme annoncé précédemment, nous remettons D à la valeur 255.

26168 NOP

Ordre blanc (l'équivalent d'un REM).

26169 INC BC

On incrémente BC d'une unité pour obtenir l'adresse de la mémoire où nous allons écrire le résultat de la compression. Comme cette incrémentation a lieu avant l'écriture, nous avons initialisé au début du programme HL à 26203, alors que le début de la zone est 26204.

26170 LD A,E

Charge la valeur de E dans A, pour permettre l'écriture dans les mémoires de la zone image compressée ; E contient la valeur commune des octets compressés.

26171 LD (BC),A

Ecrit la valeur contenue dans A dans la mémoire dont l'adresse est contenue dans BC.

26172 LD A,D

Charge la valeur de D dans A pour permettre l'écriture dans les mémoires de la zone image compressée. D contient le nombre d'octets successifs égaux.

26173 INC BC

Incrémentation de BC pour écrire à la mémoire suivante.

26174 LD (BC),A

Ecrit la valeur contenue dans A dans la mémoire dont l'adresse est contenue dans BC.

26175 JP 26138

Saut inconditionnel à l'adresse 26138 pour recommencer l'exploration de la zone image écran.

26178 INC BC

Comme nous avons fait le test de fin d'exploration avant de passer dans le morceau de programme qui écrit les résultats, quand nous sortons de la boucle d'exploration en fin d'image nous devons en inscrire le résultat dans la zone image compressée. Comme précédemment, nous incrémentons BC.

26179 LD (BC),A

Ecrit A à l'adresse contenue par BC.

28180 LD A,D

Charge D dans A.

26181 INC BC

Incrémente BC.

26182 LD (BC),A

Ecrit A à l'adresse contenue par BC.

26183 LD 26189,BC

Le nombre d'octets qui composent l'image compressée nous sera

nécessaire pour l'enregistrement sur disquette de l'image compressée, nous l'inscrivons dans les mémoires 26189 et 26190.

26187 RET

Retour au point où le programme a été appelé.

Lignes 850 et 860

Boucle en I pour charger les valeurs de notre programme de décompression en langage machine.

PROGRAMME DE DÉCOMPRESSION

42589 LD HL,49152

Charge HL avec la valeur 49152. Première adresse de la zone où nous allons reconstituer l'image. HL contiendra ensuite l'adresse courante où l'octet de l'image devra être écrit.

42592 LD BC,34395

Charge BC avec la valeur 34395. Première adresse, moins une unité, de la zone où se trouve l'image compressée. BC contiendra ensuite la première adresse du couple d'octets à décompresser.

42595 INC BC

Incrémentation de BC d'une unité.

42596 LD A,(BC)

Charge A avec la mémoire dont l'adresse est contenue dans BC. C'est la valeur commune d'une série d'octets de l'image.

42597 LD D,A

Charge D avec le contenu de A.

42598 INC BC

Incrémente BC d'une unité.

42599 LD A,(BC)

Charge A avec la mémoire dont l'adresse est contenue dans BC.

C'est le nombre d'octets successifs égaux dont on vient de transférer la valeur dans D.

42600 LD (HL),D

Ecrit la valeur de D dans la mémoire dont l'adresse est contenue dans HL.

42601 INC HL

Incrémente HL d'une unité.

42602 DEC A

Décrémente A d'une unité.

42603 JP NZ,42600

Saut conditionnel ; si Non Zéro, à l'adresse 42600. Si, au cours de la décrémentation précédente de A, celui-ci est arrivé à Zéro, l'indicateur passe à Zéro et il n'y a pas de saut, on passe à l'ordre suivant ; cela correspond au fait que l'on a écrit l'octet commun le nombre adéquat de fois dans la zone image écran. Dans le cas contraire, on doit continuer à écrire cet octet et l'on fait un saut en 42600 pour continuer l'écriture.

On va, comme pour le programme de compression, traiter la fin de décompression en testant si HL est arrivé à 65535, en suivant la même procédure.

42606 LD A,L

Charge A avec la valeur de L.

42607 XOR,255

OU exclusif avec la valeur 255.

42609 JP NZ,42595

Saut conditionnel, si Non Zéro, à l'adresse 42595.

42612 LD A,H

Charge A avec la valeur de H.

42613 XOR,255

OU exclusif avec la valeur 255.

42615 JP NZ,42595

Saut conditionnel, si Non Zéro, à l'adresse 42595 (ce qui correspond au fait que la valeur 65535 n'est pas encore atteinte). Sinon, on continue la décompression.

42618 RET

Retour au point où le programme a été appelé.

Nous passons maintenant au morceau du programme qui teste cet ensemble compression/décompression.

Ligne 1230

Appel du programme de compression.

Ligne 1240

Déplacement de l'image compressée de la zone 26204-34395 vers la zone 34396-42587, à l'aide du programme de recopie d'image mis au point précédemment.

Ligne 1250

Le résultat de la compression (taille de l'image compressée) est prélevé des mémoires où le programme de compression l'a inscrit.

Ligne 1260

Enregistrement sur disquette de l'image compressée, sous forme d'un fichier binaire dont l'adresse de départ est 34396 et la longueur, la taille de l'image compressée.

Ligne 1270

Message de fin d'enregistrement.

Ligne 1280

Remise à zéro de l'image écran et chargement de l'image compressée. Appel du programme de décompression.

Lignes 1290 et 1300

Une boucle de retard pour pouvoir examiner l'image à l'écran.

Pour compléter ce listing, voici le DUMP mémoire des deux programmes en langage machine.

PROGRAMME DE COMPRESSION

```
26131          0
26132          33
26133          0
26134          192
26135          1
26136          91
26137          102
26138          22
26139          1
26140          94
26141          35
26142          125
26143          238
26144          255
26145          194
26146          42
26147          102
26148          124
26149          238
26150          255
26151          202
26152          66
26153          102
26154          123
26155          190
26156          194
26157          57
26158          102
26159          20
26160          202
26161          54
26162          102
26163          195
```

```
26164          29
26165          102
26166          22
26167          255
26168          0
26169          3
26170          123
26171          2
26172          122
26173          3
26174          2
26175          195
26176          26
26177          102
26178          3
26179          2
26180          122
26181          3
26182          2
26183          237
26184          67
26185          77
26186          102
26187          201
26188          0
26189          95
26190          115
```

PROGRAMME DE DÉCOMPRESSION

```
42588          0
42589          33
42590          0
42591          192
42592          1
42593          91
42594          134
42595          3
42596          10
42597          87
42598          3
42599          10
```

```
42600          114
42601          35
42602          61
42603          194
42604          104
42605          166
42606          125
42607          238
42608          255
42609          194
42610          99
42611          166
42612          124
42613          238
42614          255
42615          194
42616          99
42617          166
42618          201
42619          0
```

Il ne vous reste plus qu'à introduire ce programme. L'enregistrer sur disquette. Le tester. Corriger les erreurs de frappe... et passer à la suite. Une remarque en passant : si vous avez fait des erreurs, vous courez le risque de perdre la main complètement. Dans ce cas, pour continuer, une seule possibilité : éteindre votre Amstrad et le rallumer ; le CTRL + SHIFT + ESC est habituellement inopérant.

13. LE SCROLLING D'ÉCRAN

Pour simplifier l'affichage à l'écran, l'Amstrad possède un *scrolling* d'écran par logiciel.

Examinons-en tout d'abord le fonctionnement :

Lorsque l'on écrit par un ordre PRINT non précédé d'un ordre LOCATE, l'écriture s'effectue à la ligne suivant celle où est écrit le dernier texte. Quand l'écriture est terminée, le curseur se place à la ligne suivante.

Par exemple, après un CLS dans un programme en cours d'exécution, le premier PRINT écrit sur la première ligne, le deuxième PRINT sur la deuxième ligne (à condition bien sûr que le texte du premier PRINT ne dépasse pas une ligne de longueur), et ainsi de suite. De même, lorsque l'Amstrad vous envoie un message READY ou SYNTAX ERROR, il écrit sur la ligne suivante et positionne le curseur sur la ligne d'après.

Quand on arrive à la 25^{e} ligne, le positionnement du curseur sur la ligne suivante déclenche un décalage de toutes les lignes vers le haut : le texte de la 25^{e} ligne passe à la 24^{e} ligne, celui de la 24^{e} à la 23^{e}, et ainsi de suite ; la première ligne comporte alors le texte de la deuxième ligne, et le texte de la première ligne est perdu. On a donc un scrolling avec perte du texte de la première ligne.

Dans l'Amstrad, comme nous l'avons annoncé, ce scrolling est effectué par logiciel.

Expliquons rapidement le fonctionnement de ce logiciel. Chaque ligne est représentée par des mémoires situées dans la zone d'adresses 49152 à 65535 (les mêmes bien sûr que pour l'écran graphique). Une ligne de pixels, soit 320, est représentée par 80 octets. A la mise sous tension (ou après un reset complet, un RUN ou un NEW), la première ligne de pixels correspond aux mémoires d'adresses 49152 à 49231, la deuxième ligne de pixels aux mémoires d'adresses 51200 à 51279. Pour avoir une ligne de texte, il faut huit lignes de pixels.

La première ligne de texte correspond aux mémoires d'adresses :

49152 à 49231
51200 à 51279
53248 à 53327
55296 à 55375
57344 à 57423
59392 à 59471
61440 à 61519
63488 à 63567

La deuxième ligne de texte aux mémoires d'adresses :

49232 à 49311
51280 à 51359
53328 à 53407
55376 à 55455
57424 à 57503
59472 à 59551
61520 à 61579
63568 à 63647

Pour effectuer le scrolling, la première possibilité consiste à inscrire les octets correspondant à la deuxième ligne de texte dans les mémoires correspondant à la première ligne de texte, puis ceux de la troisième ligne dans les mémoires correspondant à la deuxième ligne, et ainsi de suite. Ce n'est pas celle qu'utilise l'Amstrad.

Le logiciel qui gère l'affichage à l'écran possède une table qui lui indique l'adresse de la première mémoire qui contient le premier octet de la première ligne, et il est capable de calculer alors où se trouvent les octets adéquats pour représenter les différentes lignes, compte tenu du fait que l'on considère que la mémoire qui suit la mémoire 65535 est la mémoire d'adresse 49152. Le déclenchement du scrolling change l'adresse de la mémoire qui représente la première ligne. En pratique, l'Amstrad peut effectuer des scrollings caractère par caractère et non pas ligne par ligne.

Par conséquent, en cours d'utilisation de notre logiciel CREIMAGE, tout déclenchement du scrolling, provoqué par exemple par un message d'erreur de l'AMSDOS, perturbera notre image graphique, alors que les octets représentant notre image dans les mémoires ne seront pas modifiés. Nous allons inclure dans notre logiciel un petit programme en langage machine qui permettra de déclencher systématiquement la remise à zéro du scrolling pendant l'utilisation de CREIMAGE.

Pour cela, il suffit d'appeler un sous-programme, installé à demeure dans l'Amstrad à l'adresse 48133, en lui transférant les valeurs de remise à zéro du scrolling.

Nous implanterons ce programme à l'adresse 26610, il sera appelé par CALL 26610.

En voici les ordres en assembleur ainsi que le DUMP mémoire.

```
26110 LD HL,0        Charge la valeur 0 dans HL
26113 CALL 48133     Appel du programme de scrolling
26116 RET            Retour au programme d'appel
```

26110 33
26111 0
26112 0
26113 205
26114 5
26115 188
26116 201

Nous pouvons maintenant passer au logiciel CREIMAGE.

14. DESCRIPTION DU LOGICIEL CREIMAGE

IMPLANTATION DANS LES MEMOIRES

ADRESSES	ADRESSES	OCCUPATION
255-255	65535	Image1
0-192	49152	
123-166	42619	Décompresseur
92-166	42588	
91-166	42587	Image21 (Image2)
92-134	34396	
91-134	34395	Image22 (Image2)
92-102	26204	
91-102	26203	Recopie mémoires
79-102	26191	
78-102	26190	Compresseur
19-102	26131	
18-102	26130	Grand rectangle
5-102	26117	
4-102	26116	Scrolling
253-101	26109	
252-101	26108	Image3
253-37	9725	
252-37	9724	CREIMAGE BASIC
111-1	367	

REMARQUE PRELIMINAIRE

Le système de coordonnées de l'Amstrad ne varie pas en fonction du mode choisi : X varie toujours de 0 à 639 et Y de 0 à 399. Mais, par exemple en mode 1, il n'y a que 320 positions possibles pour X et 200 pour Y. A deux valeurs successives de X ou Y ne correspond qu'une seule position sur l'écran. Les valeurs 0 et 1 donnent le même point sur l'écran. Pour se simplifier la tâche, l'utilisateur donnera les positions sur l'écran dans un système où X variera de 0 à 319 et Y de 0 à 199, et c'est le logiciel qui fera les transformations nécessaires.

INHIBITION DES MESSAGES D'ERREUR

Quand un programme BASIC détecte une erreur en cours d'exécution, l'interpréteur envoie un message d'erreur et arrête l'exécution du programme, ce qui peut perturber le bon fonctionnement de notre logiciel CREIMAGE. Nous allons utiliser l'ordre ON ERROR GOTO, qui provoque un saut à l'étiquette que l'on inscrit à la suite de cet ordre, empêche l'envoi du message d'erreur et évite l'interruption du programme. Cet ordre ne traite que les messages d'erreur du BASIC et pas ceux de l'AMSDOS.

La place disponible en mémoire pour CREIMAGE n'étant pas très grande, plutôt que d'essayer de tester la validité des valeurs données par l'utilisateur, nous allons utiliser cet ordre pour traiter par exemple des données qui provoqueraient une division par zéro ; et, dans le sous-programme de traitement de l'erreur, nous signalerons à l'utilisateur qu'il vient de se produire une erreur de BASIC, puis nous terminerons par une boucle d'attente, suivie de l'ordre RESUME qui permet de reprendre le cours normal du programme.

```
5 ON ERROR GOTO 62000

62000 GOSUB 60030:CLS:PRINT "ERREUR BASI
C":WHILE INKEY$="":WEND:GOSUB 60040:RESU
ME 1000
```

Comme nous l'avons signalé plus haut, ON ERROR GOTO ne traite pas les messages d'erreur de l'AMSDOS, tels que DISK MISSING ou FILE NOT FOUND. Certains de ces messages n'arrêtent pas le programme (par exemple, c'est le cas lorsque l'on essaie de supprimer un fichier qui n'est pas sur la disquette), vous verrez alors apparaître un message fugitif, immédiatement effacé par CREIMAGE. D'autres messages arrêtent momentanément le programme (par exemple DISK MISSING) ; la correction de l'anomalie permet de reprendre le cours normal (mettre la disquette dans notre exemple et taper R(ETRY). Pour finir, certains messages arrêtent l'exécution (par exemple, DISK FULL).

Dans ce dernier cas, si l'on a utilisé récemment la fonction C(opie), il suffit de faire GOTO 1000 puis R(ecopie) pour rappeler IMAGE2 dans IMAGE1. Si l'on n'a pas utilisé C(opie), il faut faire un ordre à exécution immédiate :

```
CALL 26110:GOSUB 60040:GOTO 1000
```

ce qui permet de récupérer IMAGE3 dans IMAGE1 ; IMAGE3 ayant été stockée lorsque le texte correspondant à la dernière fonction appelée est apparu.

D'une manière générale, IMAGE2 et IMAGE3 ne sont jamais altérées. Pour détruire IMAGE2, il faut utiliser la fonction R(ecopie) ou commencer une compression d'image ; pour détruire IMAGE3, il faut utiliser l'appel d'une fonction. Bien sûr, NEW et le reset complet détruisent IMAGE2 et IMAGE3.

Ensuite, nous trouvons la préparation des sous-programmes en langage machine :

```
10 MEMORY 9724:MODE 1:DATA 33,0,0,205,5,
188,201,0,62,,33,,,17,,,205,68,188,201,0
,0,33,0,192,1,91,102,22,1,94,35,125,238,
255,194,42,102,124,238,255,202,66,102,12
3,190,194,57,102,20,202,54,102,195,29,10
2,22,255,0,3,123,2,122,3,2,195
20 DATA 26,102,3,2,122,3,2,237,67,77,102
,201,0,,,0,1,0,64,17,,,33,,,237,176,201,
0,33,0,192,1,91,134,3,10,87,3,10,114,35,
61,194,104,166,125,238,255,194,99,166,12
4,238,255,194,99,166,201,0
30 FOR I=26110 TO 26203:READ A:POKE I,A:
NEXT:FOR I=42588 TO 42619:READ A:POKE I,
A:NEXT:SPEED KEY 10,1:E$=" ■    "
40 B=1:I0=1:I1=24:I2=8:I3=16:P=0:BORDER
1:GOSUB 14050:LOCATE 12,1:PRINT "CREATIO
N D'IMAGES":LOCATE 15,12:PRINT "EXPLICAT
IONS":LOCATE 17,20:PRINT "TAPEZ <E>":WHI
LE INKEY$="":WEND:CLS
```

■ = Taper control G.

☐ Positionnement du HIMEM en 9724 et passage en mode 1.
☐ Les DATA correspondant aux différents sous-programmes en langage machine déjà vus : remise à zéro du scrolling, réalisation de grands rectangles (ce sous-programme sera expliqué avec la fonction ZONE), compresseur graphique, transfert de mémoires et décompresseur graphique.

Remarque

Différentes valeurs sont utilisées pour le transfert de mémoires, par

conséquent nous ne chargeons ici que les valeurs permanentes du transfert de mémoires.

☐ Lecture des DATA et écriture en mémoire à la place adéquate.

☐ La rapidité de réponse des touches est mise à sa valeur maximale pour avoir le moins de temps d'attente possible.

☐ La variable E$ servira à effacer les réponses incorrectes de l'utilisateur et à faire retentir un *ding* d'avertissement. Le *ding* est provoqué par le CTRL D qui apparaît sous la forme de " " sur le listing.

☐ Positionnement de la variable B à 1 : B représente la couleur de la bordure.

☐ Les variables I0, I1, I2 et I3 représentent les quatre couleurs des encres ; on les initialise aux valeurs de démarrage de l'Amstrad.

☐ P représente le numéro d'encre du curseur graphique ; on l'initialise sur l'encre n° 0.

☐ Message de démarrage pour indiquer quelle touche donne le résumé des fonctions de CREIMAGE.

☐ Boucle d'attente pour permettre la lecture, puis nettoyage de l'écran.

Le logiciel CREIMAGE possède plusieurs sous-programmes qui vont nous servir très souvent. Nous allons donc commencer par les décrire. En voici le listing.

```
60000 POKE 26196,92:POKE 26197,102:POKE
26199,0:POKE 26200,192:CALL 26192:RETURN
60010 POKE 26196,0:POKE 26197,192:POKE 2
6199,92:POKE 26200,102:CALL 26192:RETURN
60020 POKE 26196,254:POKE 26197,37:POKE
26199,92:POKE 26200,102:CALL 26192:RETUR
N
60030 POKE 26196,253:POKE 26197,37:POKE
26199,0:POKE 26200,192:CALL 26192:LOCATE
 1,25:PRINT FRE(""):LOCATE 1,25:PRINT "
     ":RETURN
60040 POKE 26196,0:POKE 26197,192:POKE 2
6199,253:POKE 26200,37:CALL 26192:RETURN
```

Tout d'abord, nous avons cinq sous-programmes 60000, 60010, 60020, 60030, 60040, qui vont effectuer les divers transferts d'images

qui nous sont nécessaires. Ils sont tous les cinq construits de la même manière : par une série de quatre POKE. Nous complétons le programme en langage machine de transfert d'images pour y mettre les bonnes adresses de départ et d'arrivée, puis nous appelons le transfert.

Ligne 60000

Transfert de IMAGE1 vers IMAGE2 : adresse de départ 49152, soit 0-192 ; adresse d'arrivée 26204, soit 92-102. Le nombre d'octets à transférer est constant et égal à 16384, il a été introduit au début du programme.

Ligne 60010

Transfert de IMAGE2 vers IMAGE1 : adresse de départ 26204, soit 92-102 ; adresse d'arrivée 49152, soit 0-192.

Ligne 60020

Transfert de IMAGE2 vers IMAGE3 : adresse de départ 26204, soit 92-102 ; adresse d'arrivée 9726, soit 254-37.

Ligne 60030

Transfert de IMAGE1 vers IMAGE3 : adresse de départ 49152, soit 0-192 ; adresse d'arrivée 9726, soit 254-37. De plus, nous avons ajouté un ordre PRINT FRE(" "), pour forcer l'interpréteur à "faire le ménage" dans le stockage des variables alphanumériques. Cela permet d'éviter des problèmes de MEMORY FULL ou d'écrire dans la zone de IMAGE3.

Ligne 60040

Transfert de IMAGE3 vers IMAGE1 : adresse de départ 9726, soit 254-37 ; adresse d'arrivée 49152, soit 0-192.

```
61000 LOCATE 1,24:PRINT "ENCRE= "
61010 LOCATE 8,24:INPUT "",P:IF P<0 OR P
>3 THEN LOCATE 8,24:PRINT E$:GOTO 61010
61100 LOCATE 18,24:PRINT "X =          Y
="
61110 LOCATE 22,24:INPUT "",X1:LOCATE 34
,24:INPUT "",Y1:GOSUB 60040:RETURN
```

Lignes 61000 à 61110

Acquisition d'un numéro d'encre avec test de sa validité, puis acquisition d'un couple de coordonnées X1 et Y1 pour le tracé des droites. Il n'est pas nécessaire de vérifier que ces coordonnées restent dans les limites de l'écran, car les fonctions graphiques du BASIC acceptent des coordonnées en dehors de l'écran. De plus, ces coordonnées n'influenceront pas la position de notre curseur graphique (variables X et Y).

```
61200 LOCATE 1,24:PRINT "COUL=      RAYON
=       XC=       YC=";
61210 LOCATE 7,24:INPUT "",P:IF P<0 OR P
>3 THEN LOCATE 7,24:PRINT E$:GOTO 61210
61220 LOCATE 18,24:INPUT "",R:LOCATE 27,
24:INPUT "",XC:LOCATE 36,24:INPUT "",YC:
GOSUB 60040:RETURN
```

Lignes 61200 et 61220

Rappel IMAGE3 vers IMAGE1, car ce sous-programme est appelé après avoir effectué des affichages à l'écran (c'est l'équivalent pour CREIMAGE de CLS). Acquisition du numéro d'une encre avec test de la validité et acquisition des coordonnées XC et YC du centre d'un cercle ainsi que de son rayon R sans test des valeurs (voir explication précédente). Rappel IMAGE3 vers IMAGE1.

```
61300 GOSUB 60040:LOCATE 1,23:PRINT "COU
LEUR =              X1=        Y1=        X2=
       Y2=         X3=         Y3=";
61310 LOCATE 11,23:INPUT "",P:IF P<0 OR
P>3 THEN LOCATE 11,23:PRINT E$:GOTO 6131
0
61320 LOCATE 25,23:INPUT "",X1:LOCATE 35
,23:INPUT "",Y1:LOCATE 5,24:INPUT "",X2:
LOCATE 15,24:INPUT "",Y2:LOCATE 25,24:IN
PUT "",X3:LOCATE 35,24:INPUT "",Y3:GOSUB
 60040:RETURN
```

Lignes 61300 et 61320

Rappel IMAGE3 vers IMAGE1, car ce sous-programme est appelé après avoir effectué des affichages à l'écran. Acquisition d'un numéro

d'encre avec test de la validité et acquisition des coordonnées X1, Y1, X2, Y2, X3 et Y3 des trois sommets d'un triangle sans test des valeurs (voir explication précédente). Rappel IMAGE3 vers IMAGE1.

Une autre particularité de CREIMAGE sera d'introduire dans des mémoires de la zone écran qui ne sont pas (ou plutôt peu) utilisées les informations correspondant aux couleurs des quatre encres et à la couleur de la bordure. Cela permettra de redonner à l'image les couleurs choisies au moment de sa réalisation sans faire appel à d'autres informations que celles contenues dans le fichier binaire de l'image, qu'elle soit ou non compressée.

En dehors du scrolling d'écran, nous avons vu que toutes les 2 000 mémoires il y avait 48 octets inutilisés ; nous allons choisir les quarante-huit derniers pour mettre cette information, et plus précisément la mémoire d'adresse 65529 pour la couleur de la bordure et les mémoires 65530 à 65533 pour les quatre encres. Nous aurons ensuite les précautions suivantes à prendre :

- Réinscrire ces informations chaque fois que nous ferons un CLS.
- Faire de même si on risque d'avoir un déclenchement du scrolling d'écran.

En conséquence, nous ferons ce travail d'écriture par un sous-programme qui sera appelé systématiquement à un point stratégique du programme.

■ ACQUISITION DES DEMANDES UTILISATEUR DÉPLACEMENT DU CURSEUR GRAPHIQUE ET CHOIX DU STYLO

```
1000 GOSUB 14050:BORDER B:INK 0,I0:INK 1
,I1:INK 2,I2:INK 3,I3:PLOT X,Y,P
1100 CALL 26110:A$="":WHILE A$="":A$=INK
EY$:WEND:A=ASC(UPPER$(A$)):IF A>47 AND A
<52 THEN P=A-48
1110 X=X-2*((A=243)-(A=242)):IF X<0 THEN
 X=0
1120 IF X>638 THEN X=638
1130 Y=Y-2*((A=240)-(A=241)):IF Y<0 THEN
 Y=0
```

```
1140 IF Y>398 THEN Y=398
1150 IF A<65 OR A>90 THEN 1000
1160 ON A-64 GOTO 28000,12000,23000,1600
0,29000,24000,1100,1100,14000,25000,1100
,19000,15000,26000,1100,13000,1100,21000
,18000,27000,1100,1100,1100,22000,17000,
20000
```

☐ Inscription des couleurs choisies dans la zone ECRAN1, positionnement des couleurs choisies pour les encres, affichage du curseur graphique.

☐ Remise à zéro du scrolling. Boucle d'attente d'un ordre de l'utilisateur.

☐ On calcule le code ASCII de la touche enfoncée par l'utilisateur après avoir transformé les minuscules en majuscules : variable A.

☐ Si A est compris entre 47 et 52, c'est que l'utilisateur a enfoncé la touche 0, 1, 2 ou 3, c'est donc un choix d'encre ; on calcule P = A – 48 pour mettre dans P le numéro de l'encre.

☐ Les ordres 1110 à 1150 gèrent les déplacements du curseur graphique lorsqu'ils sont effectués par les touches –>, <–, ↑ et ↓ à l'aide d'équations logiques sur la valeur des codes ASCII de ces quatre touches : –> a comme code ASCII 243 ; si A = 243, l'expression logique (A = 243) vaut – 1 (moins un) et X est augmenté de deux unités, ce qui correspond au déplacement élémentaire d'un pixel. Même raisonnement pour les autres touches. De plus, lorsque l'on arrive aux extrémités de l'écran on bloque les déplacements.

☐ Aiguillage à 26 voies en fonction de la demande de l'utilisateur. A–64 permet d'associer les valeurs 1 à la touche A, 2 à la touche B, 3 à la touche C, et ainsi de suite, et donc d'envoyer le programme à l'endroit désiré. Nous utilisons uniquement 18 voies, les autres sont libres et bouclent donc sur l'ordre 1100.

Nous passons maintenant aux différentes fonctions disponibles.

CHANGEMENT DE LA COULEUR DE BORDURE

Touche d'appel B comme Bordure.

```
12000 GOSUB 60030:LOCATE 1,24:PRINT "BOR
DURE = "
12010 LOCATE 11,24:INPUT "",B:IF B<0 OR
B>26 THEN LOCATE 11,24:PRINT E$:GOTO 120
10:ELSE GOSUB 60040:GOTO 1000
```

☐ Transfert IMAGE1 vers IMAGE3.

☐ Impression du message.

☐ Acquisition de la valeur donnée par l'utilisateur.

☐ Test de la valeur : celle-ci doit être comprise entre 0 et 26 (27 couleurs disponibles).

☐ Si la valeur n'est pas correcte, effacement de la réponse avec *ding* et attente d'une nouvelle réponse.

☐ Si la réponse est valable, on rappelle IMAGE3 vers IMAGE1 et on revient en 1000 (c'est dans l'ordre 1000 qu'on trouve l'appel au sous-programme 14050 qui inscrit la valeur de la couleur de la bordure dans les mémoires de la zone écran).

POSITION DU CURSEUR GRAPHIQUE

Touche d'appel P comme Position.

```
13000 GOSUB 60030:LOCATE 1,24:PRINT "POS
ITION : X= ";XPOS/2;" Y= ";YPOS/2
13010 WHILE INKEY$="":WEND:GOSUB 60040:G
OTO 1100
```

☐ Transfert IMAGE1 vers IMAGE3.

☐ Affichage de la position du curseur graphique à l'aide des ordres adéquats du BASIC. Rappelons que la position donnée par les ordres BASIC XPOS et YPOS correspond aux 640 et 400 positions du mode 2, donc on divise le résultat par 2.

☐ Boucle d'attente pour que l'utilisateur puisse lire l'information.

☐ Rappel de IMAGE3 vers IMAGE1 et retour en 1100.

CHOIX DES COULEURS ASSOCIEES AUX ENCRES

Touche d'appel I comme Ink.

```
14000 GOSUB 60030:LOCATE 16,22:PRINT "PA
LETTE :":LOCATE 1,24:PRINT "I0=      I1=
    I2=      I3=      (0-26)";
14010 LOCATE 4,24:INPUT "",I0:IF I0<0 OR
 I0>26 THEN LOCATE 4,24:PRINT E$:GOTO 14
010
14020 LOCATE 12,24:INPUT "",I1:IF I1<0 O
R I1>26 THEN LOCATE 12,24:PRINT E$:GOTO
14020
14030 LOCATE 20,24:INPUT "",I2:IF I2<0 O
R I2>26 THEN LOCATE 20,24:PRINT E$:GOTO
14030
14040 LOCATE 28,24:INPUT "",I3:IF I3<0 O
R I3>26 THEN LOCATE 28,24:PRINT E$:GOTO
14040:ELSE GOSUB 60040:GOTO 1000
14050 POKE 65530,I0:POKE 65531,I1:POKE 6
5532,I2:POKE 65533,I3:POKE 65529,B:RETUR
N
```

☐ Transfert IMAGE1 vers IMAGE3.

☐ Impression du message, I0, I1, I2, I3 sont les variables correspondant aux quatre encres du mode 1.

☐ Pour chacune des réponses, nous testons la validité de la réponse. Comme pour le choix de la bordure, la valeur donnée doit être comprise entre 0 et 26.

☐ Si la valeur n'est pas correcte, effacement de la réponse avec ding et attente d'une nouvelle réponse.

☐ Si la valeur est correcte, on passe à l'encre suivante avec le même test.

☐ Lorsque l'on a les quatre réponses correctes, on rappelle IMAGE3 vers IMAGE1 et on retourne en 1000 où se fait l'appel du sous-programme d'inscription des valeurs dans les mémoires de la zone écran.

☐ On trouve enfin le sous-programme qui inscrit I0 à l'adresse 65530, I1 à l'adresse 65531, I2 à l'adresse 65532, I3 à l'adresse 65533 et B (la bordure) à l'adresse 65539.

DEPLACEMENT DU CURSEUR GRAPHIQUE

Touche d'appel M comme Mouvement.

```
15000 GOSUB 60030:LOCATE 1,24:PRINT "DEP
LACEMENT":GOSUB 61100:X=2*X1:Y=2*Y1:GOTO
 1000
```

☐ Transfert IMAGE1 vers IMAGE3.

☐ Impression du message.

☐ Appel au sous-programme 61100 qui permet de recevoir un couple de coordonnées X1, Y1.

☐ On multiplie par 2 ces coordonnées pour se mettre dans le système d'unité du BASIC Amstrad.

☐ Retour en 1000 (il n'est pas nécessaire de rappeler IMAGE3 vers IMAGE1, c'est le sous-programme 61100 qui s'en charge).

TRACE D'UNE DROITE EN COORDONNEES ABSOLUES

Touche d'appel D comme Droite.

```
16000 GOSUB 60030:LOCATE 18,25:PRINT "CO
ORDONNEES ABSOLUES":GOSUB 61000:DRAW 2*X
1,2*Y1,P:GOTO 1100
```

☐ Transfert IMAGE1 vers IMAGE3.

☐ Impression du message.

☐ Appel au sous-programme 61100 qui permet de recevoir un couple de coordonnées X1, Y1.

☐ Tracé de la droite (le curseur graphique ne change pas de place pendant cette opération).

☐ Retour en 1100 (il n'est pas nécessaire de rappeler IMAGE3 vers IMAGE1, c'est le sous-programme 61100 qui s'en charge).

TRACE D'UNE DROITE EN COORDONNEES RELATIVES

Touche d'appel Y.

```
17000 GOSUB 60030:LOCATE 18,25:PRINT "CO
ORDONNEES RELATIVES":GOSUB 61000:DRAWR 2
*X1,2*Y1,P:GOTO 1100
```

☐ Transfert IMAGE1 vers IMAGE3.

☐ Impression du message.

☐ Appel au sous-programme 61100 qui permet de recevoir un couple de coordonnées X1, Y1.

☐ Tracé de la droite (le curseur graphique ne change pas de place pendant cette opération).

☐ Retour en 1100 (il n'est pas nécessaire de rappeler IMAGE3 vers IMAGE1, c'est le sous-programme 61100 qui s'en charge).

SAUVEGARDE SUR DISQUETTE

Touche d'appel S comme Sauvegarde ou Save.

Programme pour CPC 464

```
18000 GOSUB 60000:GOSUB 60030:LOCATE 1,2
3:PRINT "SAUVEGARDE NORMALE (1) COMPRESS
EE (2) :";:GOSUB 19100:INPUT "NOM :",B$:
GOSUB 60040:IF A=50 THEN CALL 26132:C=PE
EK(26189)+256*PEEK(26190)-26203:ELSE GOS
UB 60050:SAVE B$,B,26236,16384:GOTO 1803
0
18010 LOCATE 1,24:PRINT C;:IF C>8192 THE
N PRINT " IMAGE > 8K":WHILE INKEY$="":WE
ND:GOSUB 60040:GOTO 1100
18020 FOR I=0 TO 500:NEXT:POKE 26194,32:
POKE 26196,92:POKE 26197,134:POKE 26199,
92:POKE 26200,102:CALL 26192:POKE 26194,
64:GOSUB 60040:GOSUB 60050:SAVE B$,B,343
96,C:GOSUB 60000
18030 GOSUB 60060:GOTO 1100
```

```
60050 POKE 45200,101:POKE 44668,101:RETU
RN
60060 POKE 45200,37:POKE 44668,37:RETURN
```

Programme pour CPC 6128

```
18000 GOSUB 60000:GOSUB 60030:LOCATE 1,2
3:PRINT "SAUVEGARDE NORMALE (1) COMPRESS
EE (2) :";:GOSUB 19100:INPUT "NOM :",B$:
GOSUB 60040:IF A=50 THEN CALL 26132:C=PE
EK(26189)+256*PEEK(26190)-26203:ELSE MEM
ORY 16384:SAVE B$,B,26236,16384:GOTO 180
30
18010 LOCATE 1,24:PRINT C;:IF C>8192 THE
N PRINT " IMAGE > 8K":WHILE INKEY$="":WE
ND:GOSUB 60040:GOTO 1100
18020 FOR I=0 TO 500:NEXT:POKE 26194,32:
POKE 26196,92:POKE 26197,134:POKE 26199,
92:POKE 26200,102:CALL 26192:POKE 26194,
64:GOSUB 60040:MEMORY 16384:SAVE B$,B,34
396,C:GOSUB 60000
18030 MEMORY 9724:GOTO 1100
```

☐ Transfert IMAGE1 vers IMAGE2 : en effet, si la sauvegarde est sans compression, c'est IMAGE2 que l'on sauvegarde.

☐ Transfert IMAGE1 vers IMAGE3.

☐ Message pour demander si l'image doit être compressée ou non.

☐ Appel du sous-programme 19100 qui permet de tester si la réponse est bien 1 ou 2.

☐ Acquisition du nom de fichier donné par l'utilisateur.

☐ Rappel de IMAGE3 vers IMAGE1.

☐ Si A = 50 l'utilisateur demande une compression d'image, on appelle donc le programme en langage machine de compression. Ensuite, on récupère la taille de l'image compressée dans la variable C et on passe à l'instruction suivante pour terminer le travail.

☐ Si l'utilisateur n'a pas demandé de compression, on remonte le HIMEM en 16384, on effectue la sauvegarde, puis on effectue un saut en 18030 pour terminer.

☐ En 18010 et 18020 on va terminer la sauvegarde de l'image compressée : tout d'abord on affiche la taille de l'image en nombre d'octets. Si cette taille est supérieure à 8192, la sauvegarde ne peut être effectuée, on l'indique à l'utilisateur avec une boucle d'attente pour lui laisser le temps de lire, puis on rappelle IMAGE3 vers IMAGE1 et on retourne en 1100.

☐ Si la sauvegarde est possible, on effectue une boucle d'attente pour laisser à l'utilisateur le temps de lire la taille de l'image compressée. Ensuite on effectue le transfert de IMAGE22 vers IMAGE21 en indiquant les paramètres spécifiques au programme de recopie d'image. En effet, cette zone a une taille de 8 192 octets au lieu de 16 384 habituellement ; la recopie effectuée, on remet le 16384 dans le programme de recopie. On remonte le HIMEM en 16384 et on effectue la sauvegarde. On restitue ensuite IMAGE1 dans IMAGE2, car IMAGE2 a été écrasée par la compression.

☐ Pour terminer les deux cas de sauvegarde, on repositionne le HIMEM et on rappelle IMAGE3 vers IMAGE1 (en effet IMAGE1 a de nouveau été abîmée par le message donnant la taille de l'image compressée).

Les changements de HIMEM sont effectués directement par l'ordre MEMORY dans le cas du CPC 6128 ; par contre, pour le CPC 464 on fait appel aux sous-programmes 60050 et 60060 ; ces deux sous-programmes changent la valeur du HIMEM en effectuant des POKE dans les mémoires où le HIMEM est stocké par l'interpréteur. Comme signalé précédemment, l'ordre MEMORY a un bug dans le CPC 464 et ne peut donc pas être utilisé dans ce cas. Comme les adresses de stockage du HIMEM sont différentes sur ces deux types d'Amstrad, il n'est pas possible d'avoir un programme commun aux deux Amstrad.

CHARGEMENT A PARTIR DE LA DISQUETTE

Touche d'appel L comme Load.

Programme pour CPC 464

```
19000 GOSUB 60030:LOCATE 1,23:PRINT "CHA
RGEMENT NORMAL (1) DECOMPRESSE (2) :";:G
OSUB 19100:INPUT "NOM :",B$:GOSUB 60050:
LOAD B$:IF A=50 THEN CALL 42589:GOSUB 60
000:ELSE GOSUB 60010
19010 GOSUB 60060:I0=PEEK(65530):I1=PEEK
(65531):I2=PEEK(65532):I3=PEEK(65533):B=
PEEK(65529):GOTO 1000
19100 A$=INKEY$:IF A$="" THEN 19100:ELSE
 A=ASC(A$):IF A<49 OR A>50 THEN GOSUB 60
040:GOTO 1100:ELSE PRINT A$;:RETURN
```

Programme pour CPC 6128

```
19000 GOSUB 60030:LOCATE 1,23:PRINT "CHA
RGEMENT NORMAL (1) DECOMPRESSE (2) :";:G
OSUB 19100:INPUT "NOM :",B$:MEMORY 16384
:LOAD B$:IF A=50 THEN CALL 42589:GOSUB 6
0000:ELSE GOSUB 60010
19010 MEMORY 9724:I0=PEEK(65530):I1=PEEK
(65531):I2=PEEK(65532):I3=PEEK(65533):B=
PEEK(65529):GOTO 1000
19100 A$=INKEY$:IF A$="" THEN 19100:ELSE
 A=ASC(A$):IF A<49 OR A>50 THEN GOSUB 60
040:GOTO 1100:ELSE PRINT A$;:RETURN
```

☐ Transfert IMAGE1 vers IMAGE3.

☐ Message pour demander si l'image doit être décompressée ou non.

☐ Appel du sous-programme 19100 qui permet de tester si la réponse est bien 1 ou 2.

☐ Acquisition du nom de fichier donné par l'utilisateur.

☐ On remonte HIMEM en 16384 et on charge le fichier dans IMAGE2 s'il s'agit d'une image normale et dans IMAGE21 s'il s'agit d'une image compressée.

☐ Si la décompression a été demandée (A = 50), on appelle le décompresseur et on transfère IMAGE1 dans IMAGE2 puisque dans le cas de la décompression IMAGE2 n'existe pas. S'il n'y a pas de décompression, on transfère IMAGE2 dans IMAGE1.

☐ Ce sous programme possède une boucle d'attente de la frappe d'une touche, puis la vérification que la réponse est bien 1 ou 2 ; si c'est le cas, on affiche la réponse, sinon on continue d'attendre.

Même remarque que pour la fonction sauvegarde pour les changements de HIMEM.

REMPLISSAGE D'UNE ZONE

Touche d'appel Z comme Zone.

```
20000 GOSUB 60030:LOCATE 17,22:PRINT "ZO
NE":LOCATE 1,24:PRINT "RECTANGLE (1) TRI
ANGLE (2) CERCLE (3)   GRAND RECTANGLE (
4)"
20010 A$=INKEY$:IF A$="" THEN 20010 ELSE
 A=ASC(A$):GOSUB 60040:IF A<48 OR A>52 T
HEN 1100:ELSE ON A-48 GOTO 20100,20200,2
0300,20400
```

Nous proposons quatre types de zones : rectangle, triangle, cercle, grand rectangle ; nous commençons donc par l'affichage d'un menu.

☐ Transfert IMAGE1 vers IMAGE3.

☐ Affichage du menu.

☐ Boucle d'attente de la réponse de l'utilisateur, transfert de IMAGE3 vers IMAGE1 puis test de la validité de la réponse. Si la réponse n'est pas valable : retour en 1100.

☐ Aiguillage par un ON GOTO.

Rectangle

```
20100 GOSUB 60030:LOCATE 15,22:PRINT "RE
CTANGLE":LOCATE 1,24:PRINT "UNIFORME
  (1) RAYE HORIZONTAL    (2)";:LOCATE 1,
25:PRINT "RAYE VERTICAL (3) COULEURS ALT
ERNEES (4)";
20110 A$="":WHILE A$="":A$=INKEY$:WEND:A
=ASC(A$):GOSUB 60040:IF A<48 OR A>52 THE
N 1100:ELSE LOCATE 1,24:PRINT "COULEUR 1
 =    COULEUR 2 =":LOCATE 1,25:PRINT "X1
=      Y1=      X3=      Y3=";
20130 LOCATE 13,24:INPUT "",P1:IF P1<0 O
```

```
R P1>3 THEN LOCATE 13,24: PRINT E$:GOTO
20130
20140 LOCATE 28,24:INPUT "",P2:IF P2<0 O
R P2>3 THEN LOCATE 28,24:PRINT E$:GOTO 2
0140
20150 LOCATE 5,25:INPUT "",X1:LOCATE 15,
25:INPUT "",Y1:LOCATE 25,25:INPUT "",X3:
LOCATE 35,25:INPUT "",Y3:GOSUB 60040:ON
A-48 GOTO 20160,20170,20180,20190
20160 FOR I=Y1 TO Y3:MOVE 2*X1,2*I:DRAW
2*X3,2*I,P1:NEXT:GOTO 1100
20170 FOR I=Y1 TO Y3 STEP 2:MOVE 2*X1,2*
I:DRAW 2*X3,2*I,P1:MOVE 2*X1,2*(I+1):DRA
W 2*X3,2*(I+1),P2:NEXT:GOTO 1100
20180 FOR I=X1 TO X3 STEP 2:MOVE 2*I,2*Y
1:DRAW 2*I,2*Y3,P1:MOVE 2*(I+1),2*Y1:DRA
W 2*(I+1),2*Y3,P2::NEXT:GOTO 1100
20190 FOR I=Y1 TO Y3:FOR K=X1 TO X3 STEP
 2:PLOT 2*K,2*I,P1:PLOT 2*(K+1),2*I,P2:N
EXT:P=P2:P2=P1:P1=P:NEXT:GOTO 1100
```

Nous donnons de nouveau quatre possibilités et donc un menu de choix : rectangle uniforme, rectangle rayé horizontalement, rectangle rayé verticalement, rectangle avec deux couleurs alternées.

☐ Boucle d'attente du choix de l'utilisateur. Test de la validité du choix ; si le choix n'est pas accepté, retour en 1100, ce qui permet de sortir de cette fonction si on s'est trompé sur la sélection.

☐ Acquisition des deux couleurs avec test de la validité et acquisition des coordonnées des sommets inférieur gauche et supérieur droit du rectangle sans test de validité puisque les fonctions graphiques du BASIC acceptent des coordonnées en dehors des limites de l'écran.

☐ Rappel de IMAGE3 vers IMAGE1 et aiguillage sur le type de rectangle demandé.

Ligne 20160

Elle colorie le rectangle avec une seule couleur en traçant des lignes horizontales pour le remplir, la boucle est donc en Y. On a un MOVE pour se placer à gauche et un DRAW pour tracer la ligne horizontale. Retour en 1100.

Ligne 20170

Elle colorie le rectangle en traçant une ligne horizontale avec la couleur P1 et une ligne horizontale avec la couleur P2 et ainsi de suite. La boucle est donc sur Y. Retour en 1100.

Ligne 20180

Elle colorie le rectangle en traçant une ligne verticale avec la couleur P1, une ligne horizontale avec la couleur P2 et ainsi de suite. La boucle est donc en X. Retour en 1100.

Ligne 20190

Elle colorie le rectangle avec un pixel de la couleur P1, puis un pixel de la couleur P2, puis P1 et ainsi de suite. On a alors deux boucles imbriquées en X et Y avec utilisation de PLOT et permutation des couleurs. Retour en 1100.

Triangle

```
20200 GOSUB 61300:L=SQR((X3-X2)^2+(Y3-Y2
)^2):DX=(X3-X2)/2/L:DY=(Y3-Y2)/2/L:A=4*I
NT(L):MOVE 2*X1,2*Y1:I=2*X2:K=2*Y2:DRAW
I,K,P:FOR J=1 TO A:MOVE 2*X1,2*Y1:I=I+DX
:K=K+DY:DRAW I,K,P:NEXT:GOTO 1100
```

☐ Appel du sous-programme qui permet de faire l'acquisition des coordonnées des trois sommets du triangle et du numéro d'encre.

☐ La coloration du triangle se fera en traçant des droites issues du sommet X1, Y1 en direction du côté opposé. On détermine les pas de calcul DX et DY pour faire varier les coordonnées d'arrivée de cette droite et on trace les droites successives avec une boucle. Retour en 1100.

Pour certains triangles, vous constaterez que quelques pixels ne sont pas colorés ; pour les colorer il suffit habituellement de recommencer la coloration en effectuant une permutation circulaire sur l'ordre des sommets.

Cercle

```
20300 GOSUB 61200:FOR I=-PI/2 TO PI/2 ST
EP 1/R:K=2.2*R*COS(I):ORIGIN 2*XC-K,2*YC
+2*R*SIN(I):DRAW 2*K,0,P:NEXT:ORIGIN 0,0
:GOTO 1100
```

☐ Appel du sous-programme qui permet l'acquisition des coordonnées du centre du cercle, de son rayon et du numéro d'encre. On colorie le cercle en traçant des rayons successifs dont l'angle avec l'horizontale varie. Retour en 1100.

Grand rectangle

L'Amstrad possède un sous-programme qui permet de colorer des rectangles. Les coordonnées de ces rectangles sont définies comme pour l'ordre BASIC LOCATE ; de plus on définit la couleur par la valeur de l'octet qui détermine quatre pixels successifs. Le résultat sera de grands rectangles composés d'un motif se répétant tous les quatre pixels horizontaux (les verticaux seront d'une même couleur). Le programme appelé étant en langage machine, cette opération sera très rapide.

Comme pour le programme de remise à zéro du scrolling, nous écrivons un petit programme en langage machine pour transférer les valeurs adéquates et appeler le sous-programme de l'Amstrad. Il faut introduire :

La valeur de l'octet qui détermine le motif dans l'accumulateur A.
La limite gauche dans le registre H.
La limite droite dans le registre D.
La limite supérieure dans le registre L.
La limite inférieure dans le registre E.

L'adresse d'appel du sous-programme de l'Amstrad est 48196, soit 68-188.

Nous implantons notre sous-programme entre l'adresse 26118 et l'adresse 26129 :

26118 LD A
26119 valeur à introduire dans A
26120 LD HL

26121) valeur à introduire dans HL
25122)
26123 LD DE
26124) valeur à introduire dans DE
26125)
26126 CALL 48196
26129 RET

En langage machine, on obtient :

26118 62
26119 –
26120 33
26121 –
26122 –
26123 17
26124 –
26125 –
26126 205
26127 68
26128 188
26129 201

Les mémoires d'adresses 26119, 26121, 26122, 26124 et 26125 seront remplies par le programme en langage BASIC placé en fin du programme d'acquisition.

```
20400 GOSUB 60040:LOCATE 1,24:PRINT "COU
LEUR =          X1=          Y1=":LOCATE 18,
25:PRINT "X2=          Y2="
20410 LOCATE 11,24:INPUT "",P1:IF P1<0 O
R P1>255 THEN LOCATE 11,24:PRINT E$:GOTO
 20410
20420 LOCATE 22,24:INPUT "",X1:IF X1<0 O
R X1>39 THEN LOCATE 22,24:PRINT E$:GOTO
20420
20430 LOCATE 33,24:INPUT "",Y1:IF Y1<0 O
R Y1>24 THEN LOCATE 33,24:PRINT E$:GOTO
20430
```

```
20440 LOCATE 22,25:INPUT "",X2:IF X2<0 O
R X2>39 THEN LOCATE 22,25:PRINT E$:GOTO
20440
20450 LOCATE 33,25:INPUT "",Y2:IF Y2<0 O
R Y2>24 THEN LOCATE 33,25:PRINT E$:GOTO
20450
20460 GOSUB 60040:POKE 26119,P1:POKE 261
21,Y1 :POKE 26122,X1 :POKE 26124,Y2 :POK
E 26125,X2:CALL 26117:GOTO 1100
```

☐ Rappel de IMAGE3 vers IMAGE1 pour nettoyer l'écran.

☐ Acquisition de la couleur, et coordonnées du grand rectangle avec test de validité des réponses et effacement des réponses erronées (la couleur peut varier entre 0 et 255 ; les abscisses entre 0 et 39 ; les ordonnées entre 0 et 24).

☐ Rappel de IMAGE3 vers IMAGE1. Ecriture des valeurs dans les mémoires de notre programme en langage machine. Appel du sous-programme. Retour en 1100.

RECOPIE D'UNE IMAGE DANS IMAGE1

Touche d'appel R comme Recopie.

```
21000 GOSUB 60010:GOTO 1100
```

☐ Transfert d'IMAGE2 vers IMAGE1 et retour en 1100.

AIDE A L'UTILISATEUR POUR LES COULEURS

Touche d'appel X comme eXplications.

```
22000 GOSUB 60030:CLS:PRINT "0 NOIR
 9  VERT          18 VERT VIF    1 BLEU
 10 TURQUOISE     19 VERT MARIN 2 BLEU VIF
 11 BLEU CIEL     20 TURQUOI VIF3 ROUGE
 12 JAUNE         21 VERT CITRON4 MAGENTA
 13 BLANC         22 VERT PASTEL";
```

```
22010 PRINT "5 MAUVE      14 BLEU PAST  2
3 TURQU PAST 6 ROUGE VIF 15 ORANGE      2
4 JAUNE VIF  7 POURPRE   16 ROSE        2
5 JAUNE PAST 8 MAGEN VIF 17 MAGEN PAST 2
6 BLANC BRILL"
22020 PRINT "INITIAL 1-24-16-8":PRINT:PR
INT "ACTUEL ";I0;I1;I2;I3;" BORDURE :";B
22030 WHILE INKEY$="":WEND:GOSUB 60040:G
OTO 1100
```

☐ Transfert IMAGE1 vers IMAGE3.

☐ Nettoyage écran.

☐ Affichage des différentes couleurs disponibles.

☐ Rappel des couleurs initiales et affichage des couleurs actuellement utilisées.

☐ Boucle d'attente pour permettre la lecture.

☐ Rappel IMAGE3 vers IMAGE1.

☐ Retour en 1000 puisque l'on a utilisé un CLS.

COPIE D'UNE IMAGE DANS IMAGE2

Touche d'appel C comme Copie.

```
23000 GOSUB 60000:GOTO 1100
```

Transfert d'IMAGE1 vers IMAGE2 et retour en 1100.

TRACER UNE FIGURE

Touche d'appel F comme Figure.

```
24000 GOSUB 60030:LOCATE 16,22:PRINT "FI
GURE":LOCATE 1,24:PRINT    "RECTANGLE (1
) TRIANGLE (2) CERCLE (3)"
24010 A$=INKEY$:IF A$="" THEN 24010 ELSE
 A=ASC(A$)
```

```
24020 GOSUB 60040:IF A<48 OR A>51 THEN 1
100:ELSE ON A-48 GOTO 24100,24200,24300
```

Nous proposons trois types de figures : rectangle, triangle, cercle ; nous commençons donc par l'affichage d'un menu.

☐ Transfert IMAGE1 vers IMAGE3.

☐ Affichage du menu.

☐ Boucle d'attente de la réponse de l'utilisateur, transfert de IMAGE3 vers IMAGE1, puis test de la validité de la réponse. Si la réponse n'est pas valable : retour en 1100.

☐ Aiguillage par un ON GOTO.

Rectangle

```
24100 LOCATE 1,24:PRINT "COULEUR =
 X1 =        Y1 =":LOCATE 17,25:PRINT "X3
 =        Y3 ="
24110 LOCATE 11,24:INPUT "",P:IF P<0 OR
P>3 THEN LOCATE 11,24:PRINT E$:GOTO 2411
0
24120 LOCATE 22,24:INPUT "",X1:LOCATE 33
,24:INPUT "",Y1:LOCATE 22,25:INPUT "",X3
:LOCATE 33,25:INPUT "",Y3:GOSUB 60040
24130 MOVE 2*X1,2*Y1:DRAW 2*X3,2*Y1,P:DR
AW 2*X3,2*Y3:DRAW 2*X1,2*Y3:DRAW 2*X1,2*
Y1:GOTO 1100
```

☐ Acquisition du numéro d'encre et des coordonnées du sommet inférieur gauche et supérieur droit avec test de la valeur de l'encre.

Tracé du rectangle et retour en 1100.

Triangle

```
24200 GOSUB 61300
24210 MOVE 2*X1,2*Y1:DRAW 2*X2,2*Y2,P:DR
AW 2*X3,2*Y3:DRAW 2*X1,2*Y1:GOTO 1100
```

☐ Appel du sous-programme qui permet de faire l'acquisition des

coordonnées des trois sommets et du numéro d'encre avec test de la validité de l'encre.

☐ Tracé du triangle et retour en 1100.

Cercle

```
24300 GOSUB 61200:ORIGIN 2*(XC+1.1*R),2*
YC:FOR I=0 TO 2*PI+(1/R) STEP 1/R:DRAW 2
.2*R*COS(I)-2.2*R,2*R*SIN(I),P:NEXT:ORIG
IN 0,0:GOTO 1100
```

☐ Appel du sous-programme qui permet l'acquisition des coordonnées du centre du cercle, de son rayon et du numéro d'encre.

☐ Tracé du cercle à l'aide d'une boucle qui affiche le point de coordonnées X = R*COS et Y = R*SIN et retour en 1100.

☐ Transfert IMAGE1 vers IMAGE3.

Suivant la version de CREIMAGE utilisée, vous avez à votre disposition l'*une* des trois possibilités suivantes : juxtaposition d'images, symétrique d'une image (image vue dans un miroir), permutation de couleurs. Ces trois fonctions ne sont pas disponibles en même temps, car l'espace mémoire de l'Amstrad est insuffisant ; d'autre part ces fonctions sont utilisées peu fréquemment et n'ont pas besoin de coexister. Nous allons décrire les trois programmes correspondants qui utilisent les mêmes numéros d'instructions et qui seront donc introduits dans trois versions de CREIMAGE différentes.

JUXTAPOSITION D'IMAGES

Touche d'appel J comme Juxtaposition.

Comme nous aurons à charger une image supplémentaire, nous aurons besoin de déplacer le HIMEM, ce qui conduit à une version CPC 464 et une version CPC 6128.

Programme pour CPC 464

```
25000 GOSUB 60030:LOCATE 1,24:INPUT "IMA
GE A SUPERPOSER :",A$:GOSUB 60040:GOSUB
60050:LOAD A$:GOSUB 60060:GOSUB 60020:GO
SUB 60000
25100 FOR K=9726 TO 26109:IF PEEK(K)=0 T
HEN 25160 ELSE A$=BIN$(PEEK(K+39426),8):
B$=BIN$(PEEK(K),8)
25110 A0$=RIGHT$(A$,1):B0$=RIGHT$(B$,1):
A4$=MID$(A$,4,1):B4$=MID$(B$,4,1):IF B0$
<>"0" OR B4$<>"0" THEN A0$=B0$:A4$=B4$
25120 A1$=MID$(A$,7,1):B1$=MID$(B$,7,1):
A5$=MID$(A$,3,1):B5$=MID$(B$,3,1):IF B1$
<>"0" OR B5$<>"0" THEN A1$=B1$:A5$=B5$
25130 A2$=MID$(A$,6,1):B2$=MID$(B$,6,1):
A6$=MID$(A$,2,1):B6$=MID$(B$,2,1):IF B2$
<>"0" OR B6$<>"0" THEN A2$=B2$:A6$=B6$
25140 A3$=MID$(A$,5,1):B3$=MID$(B$,5,1):
A7$=MID$(A$,1,1):B7$=MID$(B$,1,1):IF B3$
<>"0" OR B7$<>"0" THEN A3$=B3$:A7$=B7$
25150 POKE K+39426,128*VAL(A7$)+64*VAL(A
6$)+32*VAL(A5$)+16*VAL(A4$)+8*VAL(A3$)+4
*VAL(A2$)+2*VAL(A1$)+VAL(A0$)
25160 NEXT:GOTO 1100
```

Programme pour CPC 6128

```
25000 GOSUB 60030:LOCATE 1,25:INPUT "IMA
GE A SUPERPOSER :",A$:GOSUB 60040:MEMORY
 16384:LOAD A$:MEMORY 9724:GOSUB 60020
25100 FOR K=9726 TO 26109:IF PEEK(K)=0 T
HEN 25160 ELSE A$=BIN$(PEEK(K+39426),8):
B$=BIN$(PEEK(K),8)
25110 A0$=RIGHT$(A$,1):B0$=RIGHT$(B$,1):
A4$=MID$(A$,4,1):B4$=MID$(B$,4,1):IF B0$
<>"0" OR B4$<>"0" THEN A0$=B0$:A4$=B4$
25120 A1$=MID$(A$,7,1):B1$=MID$(B$,7,1):
A5$=MID$(A$,3,1):B5$=MID$(B$,3,1):IF B1$
<>"0" OR B5$<>"0" THEN A1$=B1$:A5$=B5$
25130 A2$=MID$(A$,6,1):B2$=MID$(B$,6,1):
A6$=MID$(A$,2,1):B6$=MID$(B$,2,1):IF B2$
<>"0" OR B6$<>"0" THEN A2$=B2$:A6$=B6$
```

```
25140 A3$=MID$(A$,5,1):B3$=MID$(B$,5,1):
A7$=MID$(A$,1,1):B7$=MID$(B$,1,1):IF B3$
<>"0" OR B7$<>"0" THEN A3$=B3$:A7$=B7$
25150 POKE K+39426,128*VAL(A7$)+64*VAL(A
6$)+32*VAL(A5$)+16*VAL(A4$)+8*VAL(A3$)+4
*VAL(A2$)+2*VAL(A1$)+VAL(A0$)
25160 NEXT:GOTO 1100
```

Le principe de la juxtaposition est le suivant : IMAGE1 est l'image sur laquelle on va effectuer la juxtaposition et IMAGEJ l'image que l'on va juxtaposer. Si un point de IMAGEJ est de l'encre 0, on gardera pour ce point l'encre de IMAGE1, par contre si ce n'est pas l'encre 0, on adoptera l'encre de IMAGEJ (habituellement l'encre 0 constitue le fond de l'image à juxtaposer).

Traduisons maintenant cette juxtaposition au niveau des octets constituant l'image. Rappelons qu'un octet correspond à quatre points successifs (ou pixels si vous préférez), l'encre de chaque point est déterminée par un couple de bits de cet octet : le premier point correspond aux bits 0 et 4, le deuxième aux bits 1 et 5, le troisième aux bits 2 et 6, le quatrième aux bits 3 et 7. Pour une série de quatre points successifs de l'image, on a donc en présence un octet correspondant à IMAGE1 dont nous appelons les bits : A0, A1, A2, A3, A4, A5, A6, A7 et un octet correspondant à IMAGEJ dont nous appelons les bits : B0, B1, B2, B3, B4, B5, B6, B7. Si B0 et B4 sont nuls, le premier point de IMAGEJ est de l'encre 0, nous gardons alors A0 et A4 ; par contre, si B0 ou B4 ou les deux ne sont pas nuls, nous remplaçons A0 par B0 et A4 par B4. Même raisonnement pour les couples (A1,A5), (A2,A6) et (A3,A7). Passons au programme qui est la traduction exacte de ce que nous venons d'exposer :

☐ Transfert IMAGE1 vers IMAGE3.

☐ Demande du nom de fichier de l'image à juxtaposer.

☐ Rappel de IMAGE3 vers IMAGE1.

☐ Cette image est obligatoirement une image normale et pas une image compressée ; on la charge de la même manière qu'une image normale, mais on la transfère en IMAGE3 au lieu de IMAGE1, puis on transfère IMAGE1 vers IMAGE2, ce qui protège l'image obtenue avant juxtaposition et permettra d'utiliser la fonction R(ecopie) si l'on n'est pas satisfait du résultat.

☐ Boucle pour comparer les octets de IMAGE1 et IMAGE3. Si un

octet de IMAGE3 est nul, il n'y a rien à juxtaposer, ce n'est pas la peine de faire de test et on passe à l'octet suivant (cela accélère la juxtaposition).

☐ On transforme la valeur lue dans les mémoires en nombre écrit en binaire dans une variable alphanumérique.

☐ On effectue les comparaisons décrites précédemment pour décider quels bits formeront IMAGE1.

☐ On écrit finalement la valeur retenue pour l'octet de IMAGE1 après avoir calculé sa valeur en système décimal. Retour en 1100.

Remarque

Ce programme de juxtaposition est en BASIC ; il est donc assez lent (environ 5 à 10 minutes suivant la complexité de l'image à juxtaposer), nous n'en avons pas fait une version en langage machine, car il est assez peu utilisé.

SYMETRIQUE D'IMAGE

Touche d'appel J.

```
29010 PRINT "<P> : POSITION DU POINT":PR
INT "<M> : MOUVEMEMT DU POINT":PRINT "<B
> : COULEUR BORDURE":PRINT "<J> : IMAGE
SYMETRIQUE":PRINT "<I> : COULEUR DES ENC
RES":PRINT "<X> : CODES DES ENCRES"
```

```
25000 GOSUB 60000:CLS:CALL 26110:A=49152
-80:C=26203
25010 FOR I=0 TO 7:A=A+I*2048:C=C+I*2048
25020 FOR J=0 TO 24:A=A+80:C=C+80
25030 FOR K=0 TO 79
25100 A$=BIN$(PEEK(C-K),8)
25110 A0$=RIGHT$(A$,1):A4$=MID$(A$,4,1)
25120 A1$=MID$(A$,7,1):A5$=MID$(A$,3,1)
25130 A2$=MID$(A$,6,1):A6$=MID$(A$,2,1)
```

```
25140 A3$=MID$(A$,5,1):A7$=MID$(A$,1,1)
25150 POKE A+K,128*VAL(A4$)+64*VAL(A5$)+
32*VAL(A6$)+16*VAL(A7$)+8*VAL(A0$)+4*VAL
(A1$)+2*VAL(A2$)+VAL(A3$)
25160 NEXT:NEXT:A=49152-80:C=26203:NEXT:
GOTO 1100
```

Nous effectuons une symétrie du type image vue dans un miroir.

☐ Transfert de IMAGE1 vers IMAGE2, c'est IMAGE2 qui sera la source pour réaliser notre image symétrique. Nettoyage écran, remise à zéro du scrolling.

☐ Rappelons que les lignes écran sont écrites par paquets de 25, chaque ligne comporte 80 octets, et entre chaque paquet de 25 il y a 48 octets inutilisés. La symétrie va consister en une permutation des octets dans une ligne : l'octet 0 devient l'octet 79, l'octet 1 devient le 78, ..., l'octet 78 le 1, l'octet 79 le 0. A l'intérieur de l'octet, le bit 0 devient le bit 3, le bit 4 le 7, le bit 1 le 2, le bit 5 le 6, le bit 2 le 1, le bit 6 le 5, le bit 3 le 0 et le bit 7 le 4. Nous effectuons cette opération par une triple boucle pour faire le travail sur les octets d'une ligne, sur un paquet de lignes, sur l'ensemble des paquets.

☐ Permutation dans l'octet, reconstitution de l'octet, écriture de l'octet dans IMAGE1 à l'endroit adéquat.

☐ Retour en 1100.

PERMUTATION DE COULEURS

Touche d'appel J.

```
29010 PRINT "<P> : POSITION DU POINT":PR
INT "<M> : MOUVEMEMT DU POINT":PRINT "<B
> : COULEUR BORDURE":PRINT "<J> : PERMUT
ATION D'ENCRES":PRINT "<I> : COULEUR DES
 ENCRES":PRINT "<X> : CODES DES ENCRES"
```

```
25000 I3=PEEK(65531):I1=PEEK(65533):POKE
 65533,I3:POKE 65531,I1:INK 1,I1:INK 3,I
3:GOSUB 60000:FOR K=26204 TO 42587:A$=BI
N$(PEEK(K),8)
25010 A0$=RIGHT$(A$,1):A4$=MID$(A$,4,1):
IF A0$="0" AND A4$="1" THEN A0$="1":ELSE
 IF A0$="1" AND A4$="1" THEN A0$="0"
25020 A1$=MID$(A$,7,1):A5$=MID$(A$,3,1):
IF A1$="0" AND A5$="1" THEN A1$="1":ELSE
 IF A1$="1" AND A5$="1" THEN A1$="0"
25030 A2$=MID$(A$,6,1):A6$=MID$(A$,2,1):
IF A2$="0" AND A6$="1" THEN A2$="1":ELSE
 IF A2$="1" AND A6$="1" THEN A2$="0"
25040 A3$=MID$(A$,5,1):A7$=MID$(A$,1,1):
IF A3$="0" AND A7$="1" THEN A3$="1":ELSE
 IF A3$="1" AND A7$="1" THEN A3$="0"
25050 POKE K+22948,128*VAL(A7$)+64*VAL(A
6$)+32*VAL(A5$)+16*VAL(A4$)+8*VAL(A3$)+4
*VAL(A2$)+2*VAL(A1$)+VAL(A0$)
25160 NEXT:GOTO 1100
```

Il peut arriver qu'après avoir fait une image vous désiriez permuter les couleurs associées à deux encres sans changer l'image de couleur. Nous allons réaliser un programme qui effectue cette fonction en prenant pour exemple la permutation de l'encre n° 1 et l'encre n° 3. Au niveau des octets qui composent l'image, les couples de bits (0,1) qui définissent l'encre n° 1 deviendront des couples (1,1) ; les couples (1,1) qui définissent l'encre 3 deviendront des couples (0,1) qui définissent l'encre 1.

☐ On commence par permuter les valeurs qui ont été attribuées aux variables I1 et I3, et on inscrit ces nouvelles valeurs aux adresses 65531 et 65533.

☐ Transfert de IMAGE1 vers IMAGE2, c'est IMAGE2 qui sera notre source de données pour réaliser l'image.

☐ On prélève ensuite chaque octet de l'image, on le met sous forme binaire dans une variable alphanumérique et on l'analyse pour déterminer les modifications à lui faire subir.

☐ On effectue l'analyse et la transformation éventuelle des couples décrites ci-dessus.

☐ On écrit enfin le nouvel octet dans ECRAN1, après l'avoir reconstitué. Retour en 1100.

NETTOYAGE ECRAN

Touche d'appel N comme Nettoyage.

```
26000 GOSUB 60030:LOCATE 1,24:PRINT "CON
FIRMER LE NETTOYAGE ECRAN (OUI/NON)"
26010 A$=INKEY$:IF A$="" THEN 26010:ELSE
 IF UPPER$(A$)="O" THEN CLS:GOTO 1000:EL
SE GOSUB 60040:GOTO 1000
```

☐ Transfert IMAGE1 vers IMAGE3.

☐ Le nettoyage de l'écran étant une action presque irréversible (presque, car l'on peut récupérer IMAGE2 ou IMAGE3), nous envoyons un message de demande de confirmation.

☐ Boucle d'attente sur la frappe d'une touche.

☐ Si la réponse est O pour OUI, on nettoie l'écran, et on retourne en 1000 pour réinscrire les couleurs, car le CLS remet à zéro toutes les mémoires de la zone écran, y compris celles où nous inscrivons les couleurs choisies.

☐ Pour toute autre réponse, nous n'effectuons pas le nettoyage écran.

☐ Transfert IMAGE3 vers IMAGE1.

☐ Retour en 1100.

CATALOGUE DE LA DISQUETTE

Touche d'appel T comme Table des matières.

```
27000 GOSUB 60030:CLS:LOCATE 1,1:IDIR
27010 WHILE INKEY$="":WEND:GOSUB 60040:G
OTO 1000
```

☐ Transfert IMAGE1 vers IMAGE3.

☐ Nettoyage de l'écran.

☐ Appel du catalogue disquette à l'aide de l'ordre adéquat de l'AMSDOS.

☐ Boucle d'attente pour laisser l'utilisateur en prendre connaissance.

☐ Rappel IMAGE3 vers IMAGE1.

☐ Retour en 1000 pour réinscrire les couleurs qui ont été effacées par l'ordre CLS.

ANNULATION D'UN FICHIER SUR LA DISQUETTE

Touche d'appel A comme Annulation.

```
28000 GOSUB 60030:LOCATE 1,23:INPUT "ANN
ULATION DU FICHIER :",B$:PRINT "CONFIRME
R L'ANNULATION DE ";B$:PRINT "<OUI/NON>"
28010 A$=INKEY$:IF A$="" THEN 28010:ELSE
 IF UPPER$(A$)="O" THEN |ERA,@B$
28020 GOSUB 60040:GOTO 1100
```

☐ Transfert IMAGE1 vers IMAGE3.

☐ Message pour demander le nom du fichier à annuler.

☐ Lorsque le nom a été donné, on envoie un message de demande de confirmation, car une annulation de fichier est irréversible.

☐ Boucle d'attente de la réponse.

☐ Si la réponse est O, on effectue l'annulation à l'aide de l'ordre adéquat de l'AMSDOS.

☐ Quelle que soit la réponse, on rappelle IMAGE3 dans IMAGE1.

☐ Retour en 1100.

AIDE A L'UTILISATEUR

Touche d'appel E comme Explications.

```
29000 GOSUB 60030:CLS:PRINT "<S> : SAUVE
GARDE IMAGE1":PRINT "<L> : CHARGE IMAGE1
":PRINT "<C> : COPIE IMAGE1 > IMAGE2":PR
```

```
INT "<R> : COPIE IMAGE2 > IMAGE1":PRINT
"<N> : NETTOYAGE IMAGE1":PRINT "<T> : CA
TALOGUE":PRINT "<A> : ANNULATION FICHIER
"
29010 PRINT "<P> : POSITION DU POINT":PR
INT "<M> : MOUVEMEMT DU POINT":PRINT "<B
> : COULEUR BORDURE":PRINT "<J> : JUXTAP
OSITION D'IMAGES":PRINT "<I> : COULEUR D
ES ENCRES":PRINT "<X> : CODES DES ENCRES
"
29020 PRINT "<D> : DROITE (COORD ABS)":P
RINT "<Y> : DROITE (COORD RELAT)":PRINT
"<F> : FIGURE":PRINT "<Z> : ZONE":PRINT:
PRINT "<0>,<1>,<2>,<3> : STYLOS":PRINT:P
RINT "FLECHES : DEPLACEMENT"
29100 WHILE INKEY$="":WEND:GOSUB 60040:G
OTO 1000
```

☐ Transfert IMAGE1 vers IMAGE3.

☐ Nettoyage écran.

☐ Impression des différentes fonctions de CREIMAGE.

☐ Boucle d'attente pour permettre la lecture.

☐ Rappel IMAGE3 vers IMAGE1.

☐ Retour en 1000 puisque l'on a utilisé un CLS.

Pour finir, il ne vous reste plus qu'à introduire sur vos disquettes les trois CREIMAGE correspondant à votre type d'Amstrad. Sur la disquette que vous pouvez acheter avec le livre, les noms sont les suivants :

CREI464	CREI6128	CREIMAGE avec juxtaposition
CRES464	CRES6128	CREIMAGE avec symétrie
CREP464	CREP6128	CREIMAGE avec permutation

Quelques conseils d'utilisation

☐ Faites vos dessins sur une feuille de papier. Relevez les principales coordonnées. Recopiez le dessin sur une feuille transparente que vous

placerez sur l'écran à l'aide de papier adhésif.

☐ Commencez par réaliser les zones en couleurs alternées. Si la zone n'est pas un rectangle, vous pourrez ensuite la modifier en dessinant d'autres zones qui viendront mordre dessus.

☐ Faites très souvent des copies dans IMAGE2 : c'est votre protection contre les erreurs de dessin (utilisation équivalente d'une gomme).

☐ Pour déterminer les coordonnées d'un point, faites une copie dans IMAGE2, déplacez le curseur à l'aide de la fonction D(éplacement) puis avec les touches <–, –>, ↑ et ↓, et relevez la position avec la fonction P(osition), et terminez avec une recopie dans IMAGE1 pour effacer la trace du curseur.

☐ Faites souvent des sauvegardes en image compressée.

☐ *En cas de destruction d'image totale ou partielle, n'oubliez pas que IMAGE3 est disponible et récupérable si l'on n'a pas appelé de fonction.* Donc pas de précipitation et d'action intempestive : essayez de recopier IMAGE2 avec R(ecopie) car cette fonction n'altère pas IMAGE3, si IMAGE2 n'est pas bonne, faites ESC pour sortir du programme, puis CALL 26110:GOSUB 60040:GOTO1000.

☐ N'oubliez pas que si vous voulez faire apparaître du texte dans une image, avec une fenêtre par exemple, le texte s'écrira avec l'encre 1 sur fond d'encre 0. Attention donc à ne pas choisir les encres 1 et 0 toutes les deux foncées ou toutes les deux claires.

15. LES IMAGES DU JEU D'AVENTURE

Maintenant que vous avez tout compris ou presque sur le fonctionnement de CREIMAGE, nous pouvons passer à la réalisation des images pour notre jeu d'aventure, ainsi qu'à l'implantation des ordres adéquats pour faire apparaître ces images.

Chacun des lieux utilisés par le jeu d'aventure est caractérisé par une valeur de la variable C. On prend pour règle d'appeler l'image correspondante CIM + le numéro du lieu. Ainsi le boudoir de la reine est le lieu 19, l'image s'appellera CIM19.

Certaines images sont multiples en fonction des actions qui ont été faites par le joueur (dans le cas d'image double, la deuxième se verra ajouter un suffixe P). En voici la liste :

La chambre de d'Artagnan.
La chambre de d'Artagnan sans épée.
La chambre de d'Artagnan tiroir ouvert.
La chambre de d'Artagnan sans épée, tiroir ouvert.
La chambre de d'Artagnan armoire ouverte.
La chambre de d'Artagnan sans épée, armoire ouverte.
La chambre de d'Artagnan coffre ouvert.
La chambre de d'Artagnan sans épée, coffre ouvert.
Les écuries sans Bonami.
Les écuries sans Flambeau.
Les écuries sans Laflèche.
Les écuries sans Mirandole.
La chambre de la maison de la veuve.
Les gardes du cardinal dans l'estaminet.
Athos, Porthos et Aramis dans l'estaminet.
Athos, Porthos, Aramis et le patron dans l'estaminet.
La campagne française sans fleurs.
Le port de Calais avec le capitaine.
Le port de Douvres avec le capitaine.
Le champ en Angleterre sans fleurs.
La rue de Londres sans fleurs.
La secrétaire de Buckingham souriante.
La secrétaire de Buckingham déçue.
La cour de Buckingham après changement de cheval.

Enfin les images suivantes ne correspondent à aucune valeur de la variable C :

L'image de présentation.
Les soins lorsque l'on est blessé.
La mer.
L'image si l'on a perdu.
L'image si l'on a gagné.

Nous obtenons alors la table suivante de correspondance entre les lieux et les noms d'images :

Image de présentation	CIM0
Rue de Trousse Chemise	CIM1
Devant la maison de la veuve	CIM2
Devant le boudoir de la reine	CIM3
Devant la bibliothèque du duc	CIM4
Devant les écuries	CIM5
Devant la maison de l'usurier	CIM6
Devant l'estaminet	CIM7
Devant les cuisines	CIM8
Devant la salle des gardes	CIM9
Devant le restaurant	CIM10
Devant les douanes	CIM11
Devant la grange	CIM12
Devant la librairie	CIM13
Devant le secrétariat	CIM14
Devant la cour de Versailles	CIM15
Devant la cour de Buckingham	CIM16
Dans la chambre de d'Artagnan	CIM1700
Chambre de d'Artagnan sans épée	CIM1710
Chambre de d'Artagnan tiroir ouvert	CIM1701
Chambre de d'Artagnan sans épée, tiroir ouvert	CIM1711
Chambre de d'Artagnan armoire ouverte	CIM1702
Chambre de d'Artagnan sans épée, armoire ouverte	CIM1712
Chambre de d'Artagnan coffre ouvert	CIM1703
Chambre de d'Artagnan sans épée, coffre ouvert	CIM1713
Dans la maison de la veuve	CIM18
Chambre de la maison de la veuve	CIM18P
Le boudoir de la reine	CIM19
Dans la bibliothèque du duc	CIM20
Dans les écuries	CIM21
Ecuries sans Bonami	CCH1

Ecuries sans Flambeau	CCH2
Ecuries sans Laflèche	CCH3
Ecuries sans Mirandole	CCH4
Chez l'usurier	CIM22
Dans l'estaminet	CIM23
Les gardes du cardinal dans l'estaminet	CIM23G
Athos, Porthos et Aramis dans l'estaminet	CIM23D
Athos, Porthos, Aramis et le patron dans l'estaminet	CIM23DP
Dans les cuisines	CIM24
Dans la salle des gardes	CIM25
Dans le restaurant	CIM26
Dans les douanes	CIM27
Dans la grange	CIM28
Dans la librairie	CIM29
Dans le secrétariat du duc	CIM30
La secrétaire de Buckingham souriante	CIM30F
La secrétaire de Buckingham déçue	CIM30P
Dans la cour de Versailles	CIM31
Dans la cour de Buckingham	CIM32
La cour de Buckingham après changement de cheval	CIM32P
La campagne française	CIM33
La campagne française sans fleurs	CIM33P
Le port de Calais	CIM34
Le port de Calais avec le capitaine	CIM34P
Le port de Douvres	CIM35
Le port de Douvres avec le capitaine	CIM35P
Le champ en Angleterre	CIM36
Le champ en Angleterre sans fleurs	CIM36P
Rue de Londres	CIM37
La rue de Londres sans fleurs	CIM37P
Le relais de la Jument bleue	CIM38
Les soins lorsque l'on est blessé	CIMSOIN
La mer	CIMMER
Image si l'on a perdu	CIMPRIS
Image si l'on a gagné	CIM39

D'autre part, notre jeu avec ses images va occuper quatre faces de disquette. Voici la répartition des images par face de disquette pour obtenir le moins de changements de disquette en cours de jeu :

FACE 1	FACE 2	FACE 3	FACE 4
CIM0	CIM1	CIM2	CIM4
	CIM5	CIM3	CIM10
	CIM6	CIM8	CIM11
	CIM7	CIM9	CIM12
	CIM170	CIM15	CIM13
	CIM171	CIM18	CIM14
	CIM172	CIM18F	CIM16
	CIM173	CIM19	CIM20
	CIM174	CIM24	CIM26
	CIM175	CIM25	CIM27
	CIM176	CIM31	CIM28
	CIM21	CIM33	CIM29
	CCH1	CIM33P	CIM30
	CCH2	CIM34	CIM30P
	CCH3	CIM34P	CIM30F
	CCH4	CIM38	CIM32
	CIM22	CIMSOIN	CIM35
	CIM23	CIMPRIS	CIM35P
	CIM23G	CIMMER	CIM36
	CIM23D	CIM39	CIM36
	CIM23DP		CIM37
	CIMSOIN		CIM37P
	CIMPRIS		CIMPRIS
			CIMMER

Nous verrons plus loin que nous installerons un test dans le programme qui vérifiera avant chargement d'une image que la bonne disquette est bien dans le lecteur de disquette ; de plus, au changement de disquette le test permettra aussi d'afficher quelle disquette doit être introduite.

Nous vous donnons en annexe les différentes images que nous avons réalisées, ainsi que les couleurs associées aux quatre encres et à la bordure.

Passons maintenant aux morceaux du programme qui sont spécifiques du chargement des images.

En tête de programme nous avons bien sûr le positionnement du HIMEM en 34380, le chargement de la remise à zéro du scrolling entre 34381 et 34387, puis le chargement du décompresseur entre 42588 et 42619. Ces opérations sont faites dans le programme DEBUT, car le fait de charger un autre programme par LOAD ou RUN ou CHAIN

ou MERGE ne change pas le HIMEM, et ne remet pas à zéro les mémoires situées au-dessus du HIMEM.

Ligne 1000

Nous installons un GOSUB 64000 et c'est là que nous traiterons le chargement des images.

```
64000 IM$=STR$(C):IM$="CIM"+RIGHT$(IM$,L
EN(IM$)-1)
64010 IF C=17 THEN IM$=IM$+RIGHT$(STR$(C
10%),1)+RIGHT$(STR$(C4%+2*C6%+3*C8%),1):
ELSE IF C=21 AND C11%<>0 THEN R$=STR$(C1
1%):IM$="CCH"+RIGHT$(R$,LEN(R$)-1)
64020 IF (C=32 AND C11%=6) OR ((C=34 OR
C=35) AND C35%=1) OR (C=33 AND C36%=1) O
R (C=36 AND C36%=2) OR (C=37 AND C36%=3)
 THEN IM$=IM$+"P"
64100 LOAD "D":I=PEEK(65529)
64110 IF (C=1 OR C=5 OR C=6 OR C=7 OR C=
17 OR C=21 OR C=22 OR C=23) AND I<>2 THE
N I=2:GOTO 64400:ELSE IF (C=2 OR C=3 OR
C=8 OR C=9 OR C=15 OR C=18 OR C=19 OR C=
24 OR C=31 OR C=33 OR C=34 OR C=38) AND
I<>3 THEN I=3:GOTO 64400
64130 IF (C=4 OR C=10 OR C=11 OR C=12 OR
 C=13 OR C=14 OR C=16 OR C=20 OR C=26 OR
 C=27 OR C=28 OR C=29 OR C=30 OR C=32 OR
 C=35 OR C=36 OR C=37) AND I<>4 THEN I=4
:GOTO 64400:ELSE LOAD IM$
64200 INK 0,0:INK 1,0:INK 2,0:INK 3,0:CA
LL 34381:CALL 42589:INK 0,PEEK(65530):IN
K 1,PEEK(65531):INK 2,PEEK(65532):INK 3,
PEEK(65533):BORDER PEEK(65529):WINDOW £0
,1,40,22,25:CLS:RETURN
64400 CLS:PRINT "METTEZ LA FACE :";I:WHI
LE INKEY$="":WEND:CLS:GOTO 64100
```

Ligne 64000

Nous effectuons la concaténation du mot CIM avec le numéro du lieu qui est donné par la variable C, pour obtenir le nom de l'image à charger ; ce nom est dans la variable IM$.

Ligne 64010

Nous traitons d'abord dans cette instruction le cas de la chambre de d'Artagnan : toutes les images correspondant à ce lieu commencent par le radical CIM17 car C = 17 ; nous calculons le suffixe en fonction des conditions : avoir ouvert le tiroir, avoir ouvert l'armoire, avoir ouvert le coffre et avoir l'épée. Nous effectuons ensuite la concaténation avec IM$. Lorsque le joueur prend la cape dans l'armoire après l'avoir ouverte, la variable correspondant à la condition armoire ouverte est remise à zéro pour revenir à la position armoire fermée ; on revient alors à l'image armoire fermée. De même pour le coffre et pour le tiroir. Puis nous traitons le cas de l'écurie : à partir du moment où le joueur a pris un cheval le radical de l'image est CCH, le numéro du cheval est contenu dans la variable C11 %, on effectue la concaténation de CCH avec la valeur de C11 %.

Ligne 64020

Nous traitons dans cette instruction la plupart des lieux qui ont deux images en fonction d'une condition, dans ce cas la deuxième image, celle qui dépend de la condition, se voit ajouter le suffixe P par concaténation :

C = 32 Cour de Buckingham, C11 % = 6 avoir changé de cheval.
C = 34 Port de Calais, C35 % = 1 avoir appelé le capitaine.
C = 35 Port de Douvres, C35 % = 1 avoir appelé le capitaine.
C = 33 Campagne anglaise, C36 % = 1 avoir cueilli les fleurs.
C = 36 Champ français, C36 % = 2 avoir cueilli les fleurs.
C = 37 Rue de Londres, C36 % = 3 avoir acheté les fleurs.

Ligne 64100

Test de la disquette : nous installerons sur chaque disquette un fichier binaire correspondant à la mémoire d'adresse 65529 dont le nom sera D et dans lequel nous inscrirons le numéro de la disquette. Nous chargeons ici ce fichier, après chargement nous avons donc dans la

mémoire 65529, le numéro de la disquette que nous copions dans la variable I.

Lignes 64110 à 64130

Nous avons ici la table de correspondance entre les numéros de faces de disquette et les numéros des lieux ; si la variable I ne vérifie pas la correspondance, nous mettons le numéro de la disquette nécessaire dans la variable I et nous effectuons un saut à l'instruction 64400 ; s'il y a correspondance, nous chargeons l'image.

Ligne 64200

On met toutes les encres à la couleur noire pour effacer l'écran pendant les manipulations sans perdre l'image. On appelle la remise à zéro du scrolling puis le décompresseur, on positionne les différentes encres, on établit la fenêtre pour les messages et on nettoie cette fenêtre. Retour au point où le sous-programme a été appelé.

Ligne 64400

Affichage de la face de disquette demandée suivi d'une boucle d'attente pour permettre la lecture et le changement de disquette.

Nous allons traiter maintenant les cas particuliers :

☐ Chargement de CIMMER : en 1540 on trouve l'instruction qui traite l'embarquement et le passage en Angleterre, nous ajoutons IM$ = "CIMMER" et nous appelons le sous-programme de chargement d'image en 64100 puisque nous n'avons pas besoin de déterminer le nom de l'image. De même pour le passage en France, nous ajoutons IM$ = "CIMMER" dans l'instruction 1590.

```
1540 C13%=C13%-300:PRINT "TOUT LE MONDE
EMBARQUE, ON LEVE L'ANCRE ET...":IM$="CI
MMER":GOSUB 64100:GOSUB 45000

1590 C15%=C15%-30:PRINT "EVERYBODY IS ON
 BOARD AND...":IM$="CIMMER":GOSUB 64100:
GOSUB 45000:C=34:C33%=1:C35%=0:C26%=0
```

☐ Traitement de l'estaminet : dans ce lieu il y a quatre images ; nous traitons les trois images supplémentaires en 9530, 9550 et 12040. Dans chacune de ces instructions nous ajoutons l'ordre correspondant à l'image adéquate :

IM$ = "CIM23G" dans 9530
IM$ = "CIM23D" dans 9550
IM$ = "DIM23DP" dans 12040

ordre que nous faisons suivre d'un GOSUB 64100 comme précédemment.

```
9530 IF NM=30 THEN IM$="CIM23G":GOSUB 64
100:PRINT "CE SONT LES GARDES DU CARDINA
L.":PRINT "ILS SONT NOMBREUX ET BIEN ARM
ES":GOTO 10


9550 IM$="CIM23D":GOSUB 64100:PRINT "CE
SONT ATHOS, PORTOS ET ARAMIS.":PRINT "AP
RES EXPLICATIONS,":PRINT "ILS DECIDENT D
E VOUS SUIVRE":C16%=1:GOTO 10



12040 IM$="CIM23DP":GOSUB 64100:PRINT "V
OS AMIS ONT BU POUR 50 PISTOLES":C17%=1:
GOTO 10
```

☐ Traitement du secrétariat du duc : dans ce lieu nous avons trois images que nous traitons de la même manière en complétant les instructions 6690 et 6700.

IM$ = "CIM30P" dans 6690
IM$ = "CIM30F" dans 6700

ordre que nous faisons suivre d'un GOSUB 64100 comme précédemment.

```
6690 IF C36%=1 THEN IM$=IM$+"P":GOSUB 64
100:PRINT "THEY'VE TURNED TO FADE.":PRIN
T "SECRETARY IS VERY HURTED":GOTO 10
6700 IF C36%>1 THEN C38%=1:IM$="CIM30F":
GOSUB 64100:PRINT "GRATEFULL, SHE GIVES
YOU A TENDER PECK. THE DUKE IS WAITING F
OR YOU":PRINT "IN HIS LIBRARY":C36%=0:GO
TO 10
```

☐ Traitement de l'écurie : lorsque l'on prend un cheval on ne repasse pas à l'instruction 1000, on traite donc le changement d'image au moment où l'on traite l'action prendre un cheval, il suffit d'ajouter dans l'instruction 4185 GOSUB 64000.

```
4185 IF C11%<>0 THEN GOSUB 64000:PRINT "
VOUS AVEZ ";NO$(NM):GOTO 10
```

☐ Chez la veuve : même problème, si l'on embrasse la veuve on ne repasse pas en 1000 ; on modifie donc l'instruction 16510 qui traite le verbe *embrasser* en lui ajoutant : IM$ = "CIM18P" : GOSUB 64100.

```
16510 IM$="CIM18P":GOSUB 64100:PRINT "EL
LE VOUS ENTRAINE DANS SA CHAMBRE...":PRI
NT "VOUS PASSEZ UNE FOLLE NUIT D'AMOUR..
.":PRINT "...TROP TARD !":GOTO 50000
```

☐ On peut attaquer les gardes du cardinal dans deux lieux, on traite l'apparition de l'image CIMSOIN, de l'image CIMPRIS ou le retour dans l'entrée de l'estaminet, dans les instructions du verbe *attaquer* (6030 à 6070) en ajoutant :

IM$ + "CIMPRIS" : GOSUB 64100 dans 6030
GOSUB 64000 dans 6040
GOSUB 64000 dans 6050
IM$ = "CIMSOIN" : GOSUB 64100 dans 6060
IM$ + "CIMPRIS" : GOSUB 64100 dans 6070

```
6030 IF C10%<>1 THEN IM$="CIMPRIS":GOSUB
 64100:PRINT "VOUS N'AVEZ PAS D'ARME ET
```

```
ILS VOUS FONT ARRETER...":GOTO 50000
6040 GOSUB 64000:PRINT "GRACE A VOTRE EP
EE,":PRINT "ILS NE PEUVENT QUE VOUS REPO
USSER":GOTO 1000
6050 IF C10%=1 AND C16%=1 THEN GOSUB 640
00:PRINT "ILS VOUS REPOUSSENT":GOTO 1000
6060 IF C10%=1 OR C16%=1 THEN C0%=C0%+15
:IM$="CIMSOIN":GOSUB 64100:PRINT "ILS VO
US BLESSENT":PRINT "ET VOUS PERDEZ DU TE
MPS A VOUS SOIGNER":GOTO 1000
6070 IM$="CIMPRIS":GOSUB 64100:PRINT "SE
UL ET SANS ARMES,":PRINT "VOUS VOUS FAIT
ES ARRETER...":GOTO 50000
```

☐ Nous avons traité le cas de la chambre de d'Artagnan à l'ordre 64010, et nous avons signalé que si le joueur prend l'objet situé dans l'armoire, l'armoire se referme, de même pour le coffre et le tiroir. Mais il reste le cas où le joueur referme par exemple l'armoire sans prendre la cape. Pour traiter ce cas nous ajoutons en 5580 un GOSUB 64000.

```
5580 GOSUB 64000:PRINT "VOILA C'EST FERM
E":GOTO 10
```

Nous donnons maintenant le petit programme qui permet de créer le fichier D qui contient le numéro de la disquette.

```
10 INPUT "D= ",D
20 POKE 65529,D
30 SAVE "D",B,65529,1
```

Ce programme n'appelle pas de commentaires particuliers : vous mettez la variable D à 1 puis vous insérez la face 1 et vous exécutez le programme, puis D à 2 la face 2 idem, puis 3, puis 4.

16. L'ARCHITECTURE DU JEU

Une fois que ce programme de jeu d'aventure est réalisé, il n'y a pas de problèmes particuliers de taille de mémoire pour l'Amstrad, tant que l'on n'a pas d'image. Si l'on ajoute les images en ne faisant pas de traitement spécial, c'est-à-dire que l'on charge les images directement dans la mémoire écran à partir de la disquette, tout continue à bien se passer, mais l'apparition des images à l'écran se fait par lignes entrelacées, ce qui n'est pas très agréable, comme nous l'avons déjà signalé. D'autre part il est nécessaire d'utiliser sept disquettes. Nous avons donc opté pour la compression d'image. Dans ce cas, comme nous l'avons vu précédemment, nous sommes obligés de baisser le HIMEM de 8K octets, ce qui ne laisse plus assez de place pour le programme d'aventure.

Il est donc nécessaire de le couper en trois parties : la première, que nous appellerons DEBUT, permettra de charger les programmes en langage machine, de faire apparaître le texte d'introduction et la première image. Puis nous appellerons AVENT le programme qui comportera tout le jeu d'aventure sauf sa fin. Cette fin de jeu s'appellera bien sûr FIN et sera chargée lorsque le joueur aura perdu ou gagné. Celle-ci comportera l'image de fin gagnante ainsi que la musique pour les deux fins possibles. Pour détecter si c'est une fin heureuse ou malheureuse, nous allons transmettre du programme AVENT au programme FIN une valeur 0 ou 1 respectivement à chacune des deux situations. Il n'est pas possible dans le BASIC de l'Amstrad de transmettre la valeur d'une variable d'un programme à un autre, nous allons donc faire un POKE de la valeur adéquate dans la mémoire 65528 (comme pour les couleurs des encres des images) au niveau du programme AVENT, et nous récupérons cette valeur par un PEEK au début du programme FIN, ce qui nous permettra de jouer la bonne musique.

Le programme sera donc lancé par un RUN "DEBUT" situé sur la face 1, on trouvera ensuite sur la face 1 le programme AVENT et l'image CIM0. Le programme FIN sera situé sur les trois autres faces, car le jeu peut se terminer en plusieurs endroits. De même l'image CIMPRIS accompagnera le programme FIN. L'image CIM39 : "On a gagné" se trouve uniquement sur la face 3.

A. TABLE DES COULEURS DES IMAGES

IMAGE	BORDURE	ENCRE 1	ENCRE 2	ENCRE 3	ENCRE 4
CIM0	0	0	7	23	26
CIM1	10	26	0	3	24
CIM2	1	0	24	11	3
CIM3	0	0	25	6	15
CIM4	2	0	25	16	4
CIM5	0	3	24	11	9
CIM6	0	0	24	14	3
CIM7	0	0	24	11	3
CIM8	0	0	13	25	3
CIM9	0	0	13	25	3
CIM10	0	0	24	11	6
CIM11	0	0	26	14	15
CIM12	0	0	13	1	15
CIM13	1	0	13	3	25
CIM14	2	0	25	22	10
CIM15	0	0	24	25	13
CIM16	0	0	26	7	24
CIM1700	2	0	25	15	20
CIM1710	2	0	25	15	20
CIM1701	2	0	25	15	20
CIM1711	2	0	25	15	20
CIM1702	2	0	25	15	20
CIM1712	2	0	25	15	20
CIM1703	2	0	25	15	20
CIM1713	2	0	25	15	20
CIM18	0	0	26	16	15
CIM18P	0	0	26	16	15
CIM19	0	0	26	6	15
CIM20	0	0	26	6	9
CIM21	1	0	26	15	24
CCH1	1	0	26	15	24
CCH2	1	0	25	15	24
CCH3	1	0	25	15	24
CCH4	1	0	25	15	24

IMAGE	BORDURE	ENCRE 1	ENCRE 2	ENCRE 3	ENCRE 4
CIM22	0	6	24	26	18
CIM23	5	0	24	6	26
CIM23G	5	0	24	6	26
CIM23D	5	0	24	6	26
CIM23DP	5	0	24	6	26
CIM24	2	0	24	6	26
CIM25	0	0	26	2	6
CIM26	0	0	26	6	11
CIM27	1	0	26	6	2
CIM28	0	0	24	13	15
CIM29	1	0	25	6	11
CIM30	1	0	26	6	11
CIM30F	1	0	26	6	11
CIM30P	1	0	26	6	11
CIM31	0	0	26	3	25
CIM32	0	0	26	20	24
CIM32P	0	0	26	20	24
CIM33	0	3	15	14	9
CIM33P	0	3	15	14	9
CIM34	2	0	24	1	26
CIM34P	2	0	24	1	26
CIM35	2	0	25	1	26
CIM35P	2	0	25	1	26
CIM36	0	3	25	11	18
CIM36P	0	3	25	11	18
CIM37	0	0	26	6	24
CIM37P	0	0	26	6	24
CIM38	0	0	25	11	3
CIMSOIN	1	0	25	26	6
CIMMER	0	0	26	1	2
CIMPRIS	0	0	7	23	26
CIM39	13	0	26	2	6

B. LES IMAGES

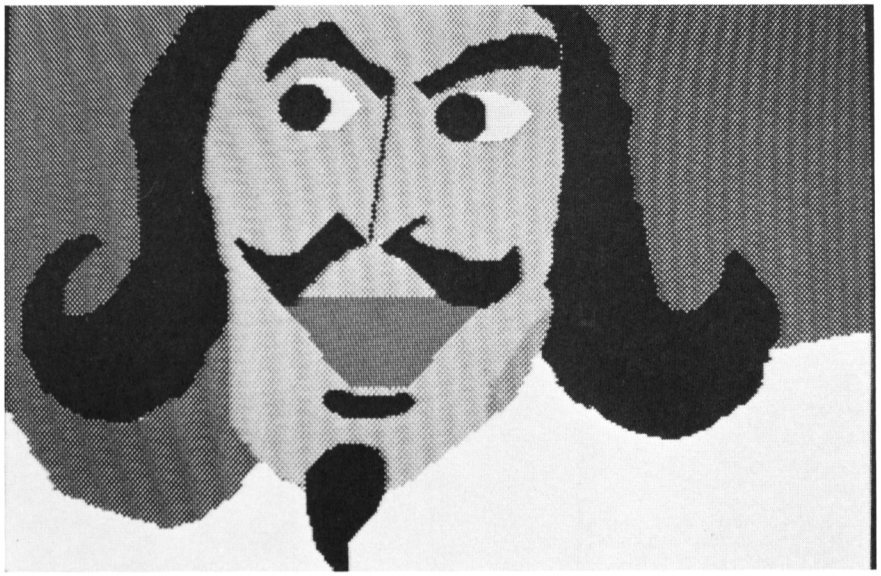
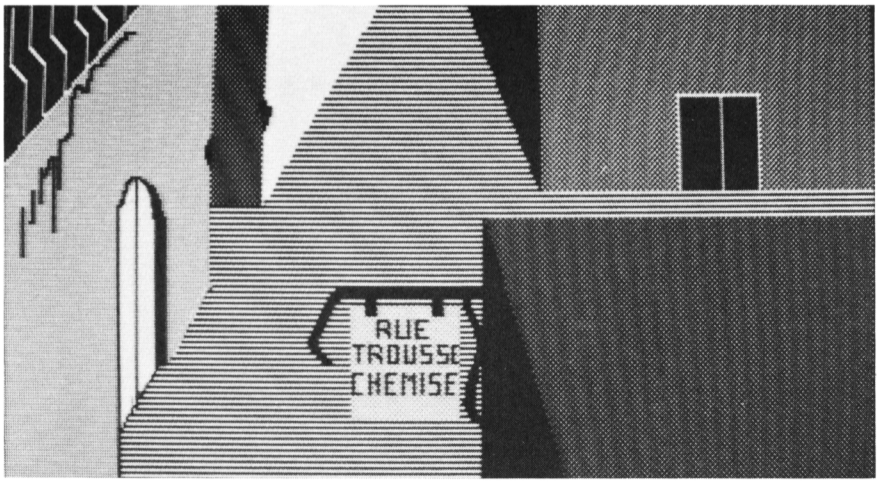
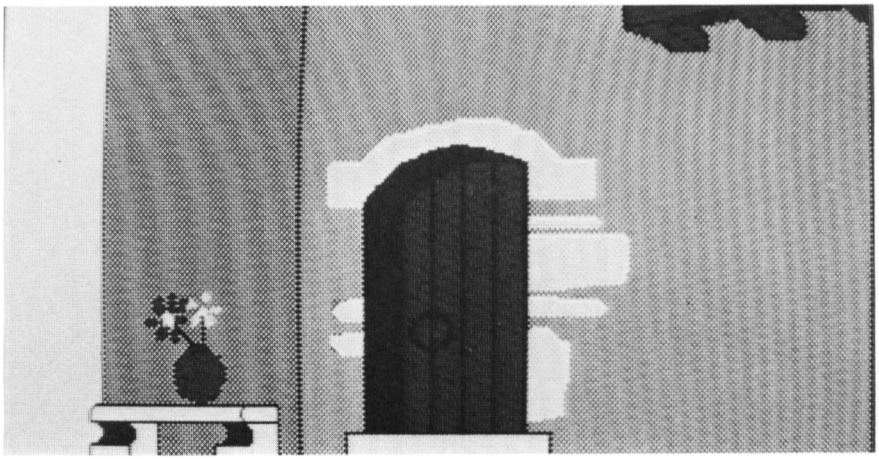


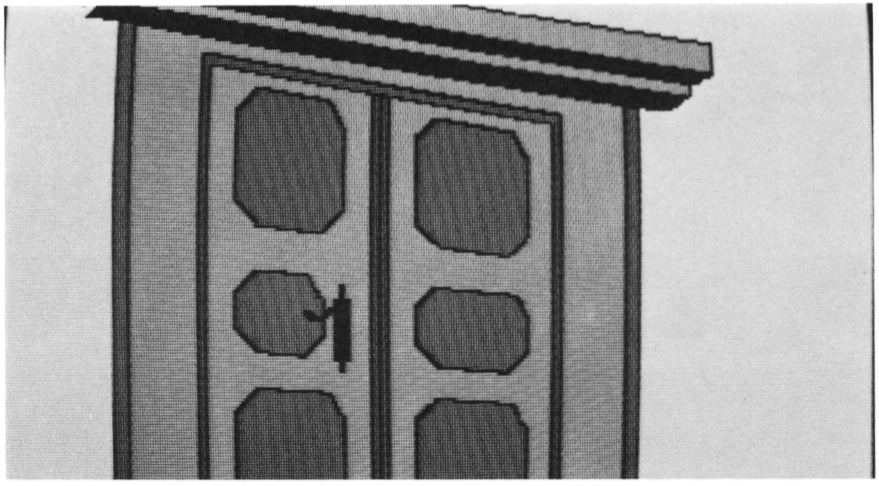
Image de présentation



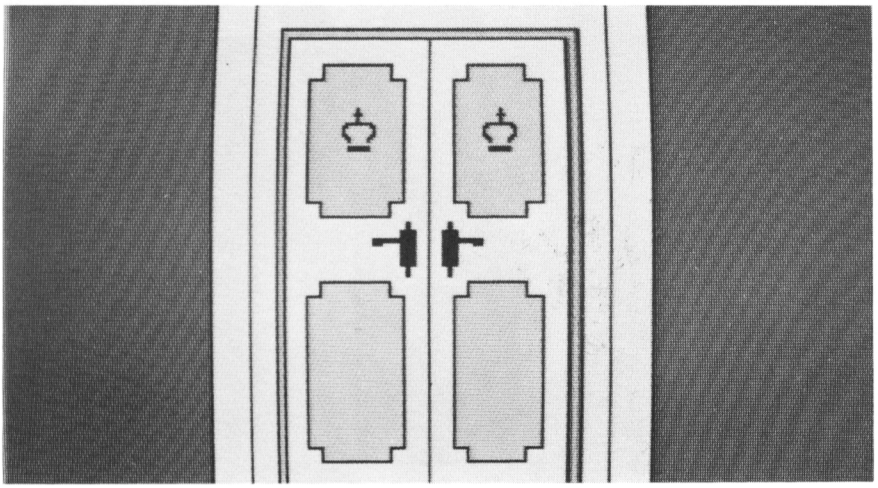
Rue de Trousse Chemise



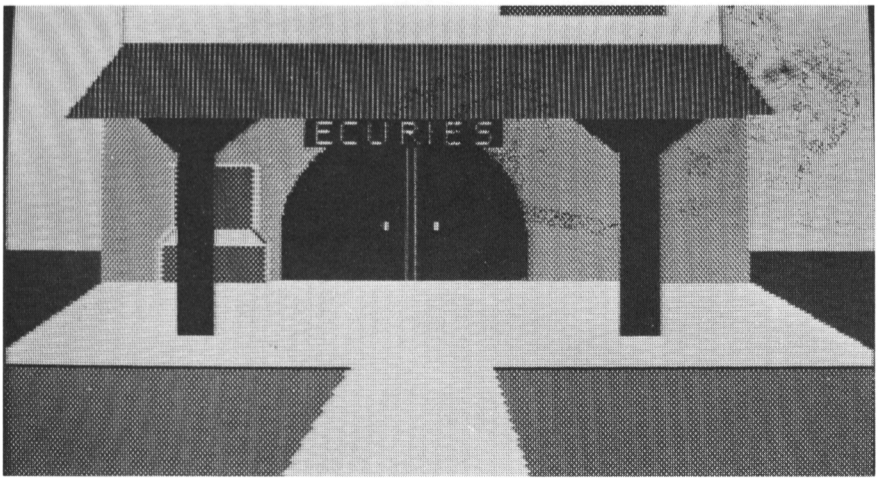
Devant la maison de la veuve



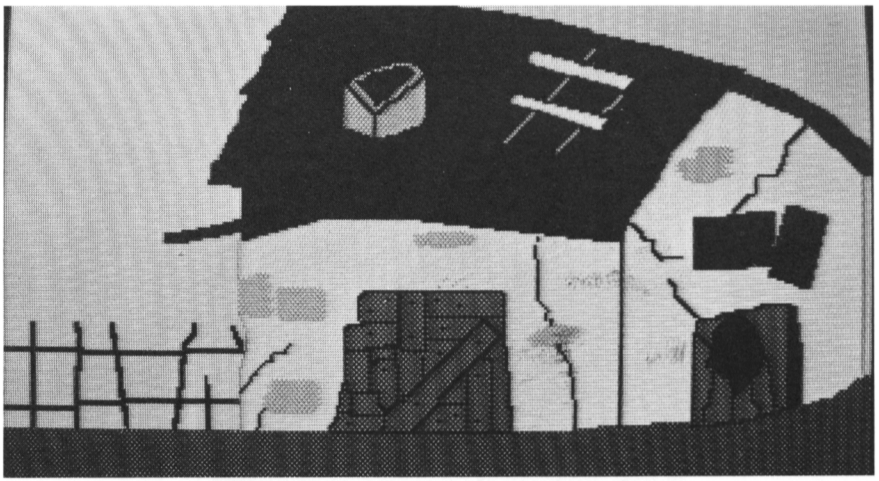
Devant le boudoir de la reine



Devant la bibliothèque du duc

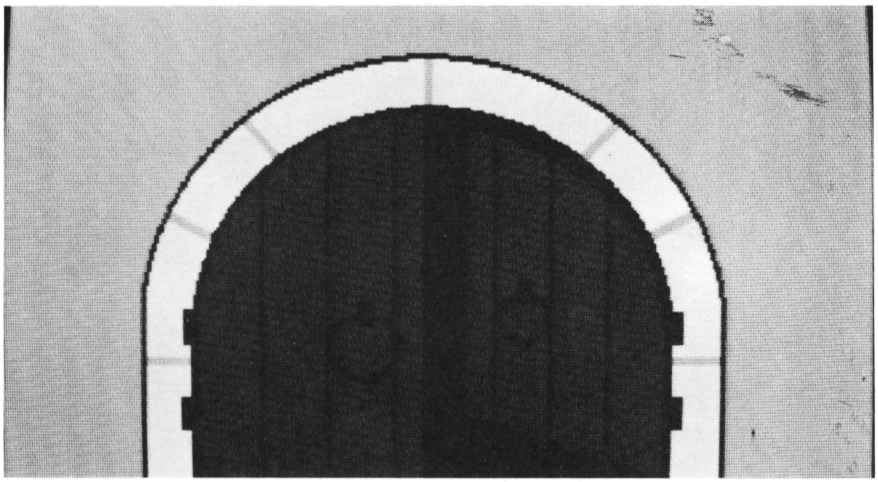


Devant les écuries

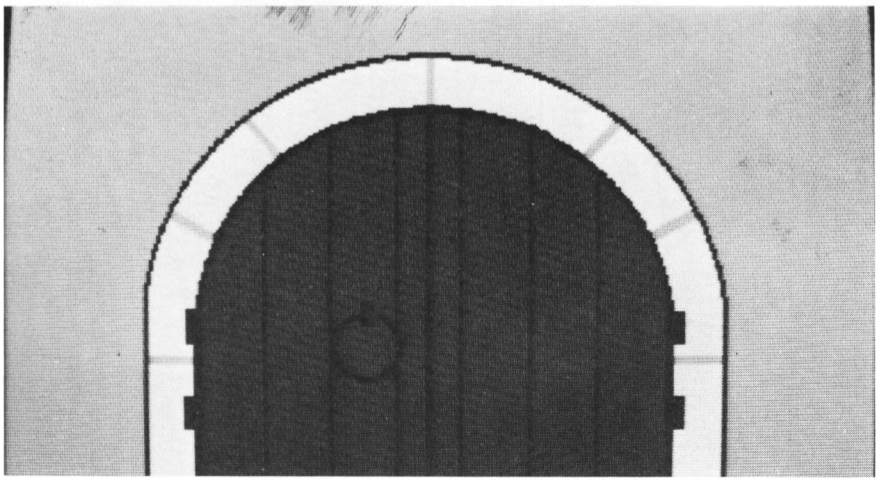


Devant la maison de l'usurier

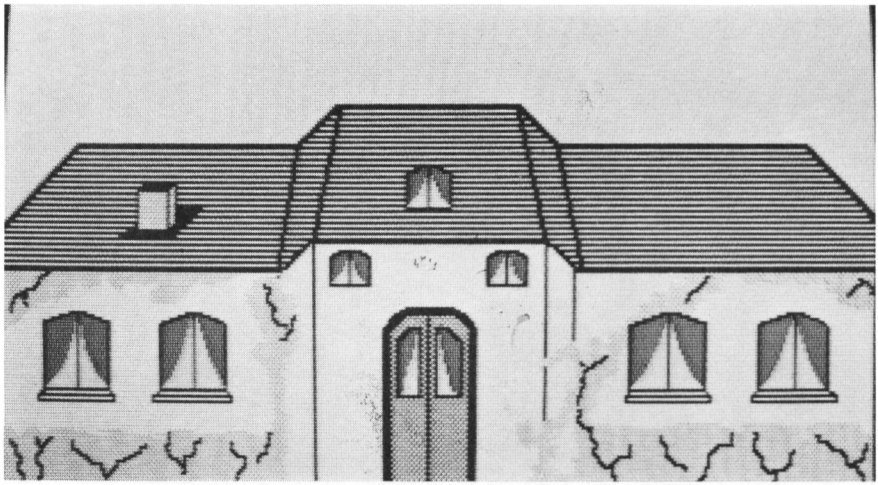
Devant l'estaminet



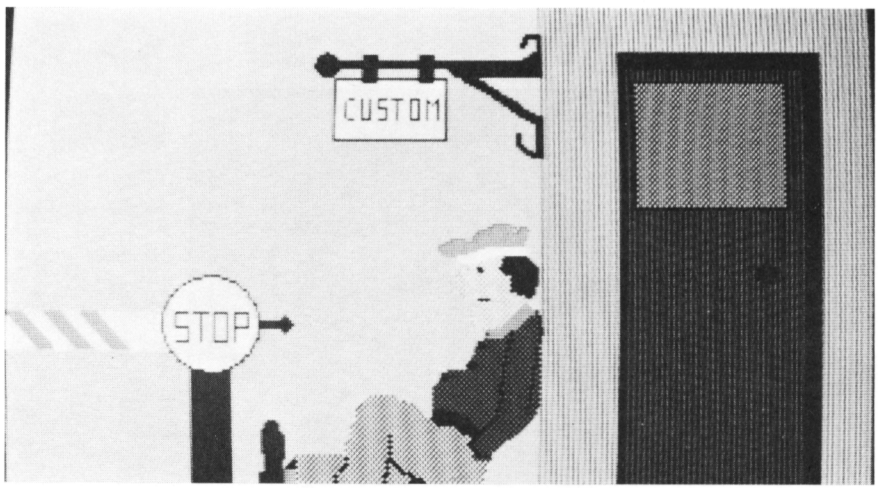
Devant les cuisines



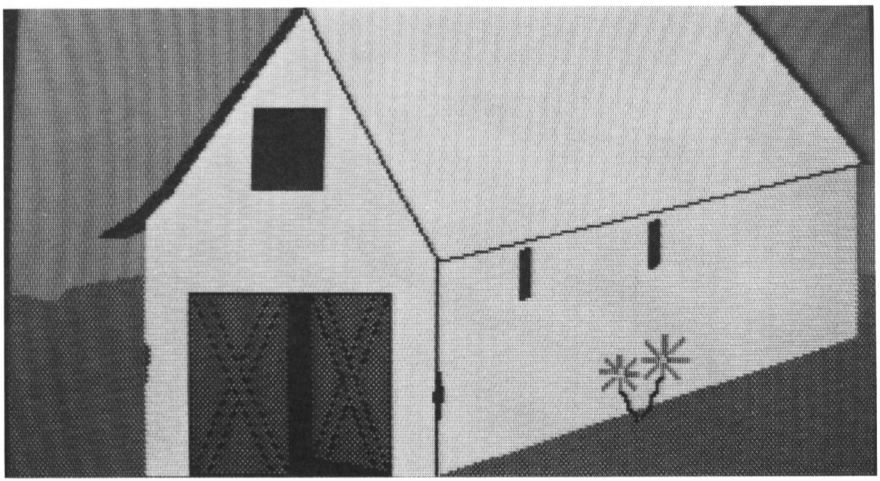
Devant la salle des gardes



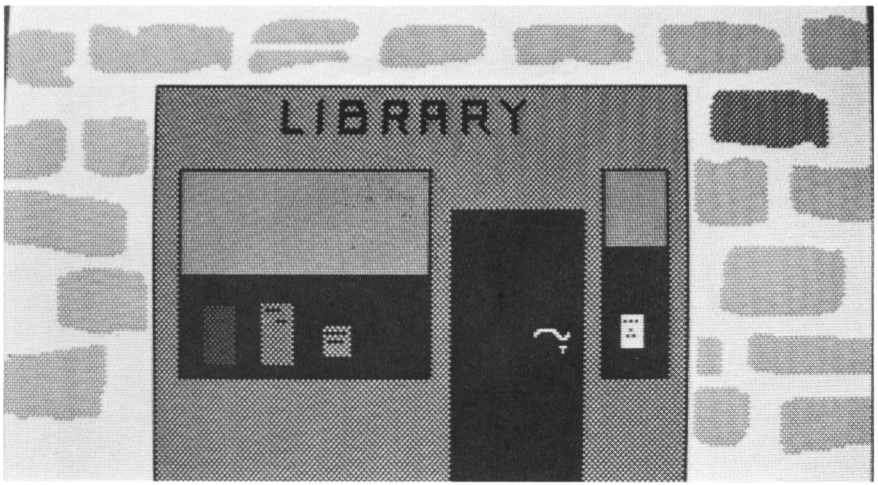
Devant le restaurant



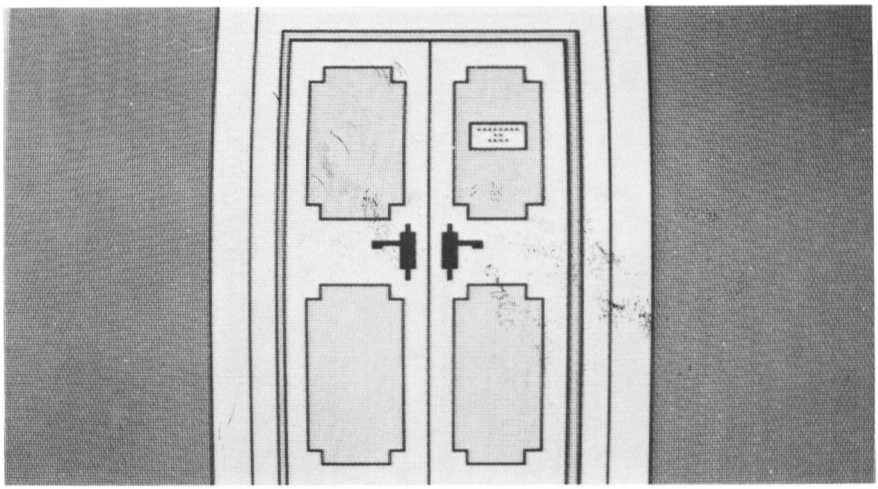
Devant les douanes



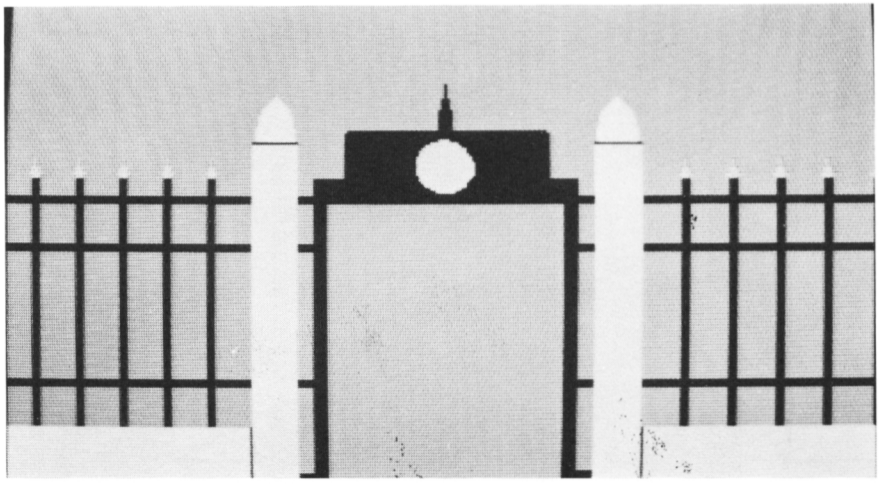
Devant la grange



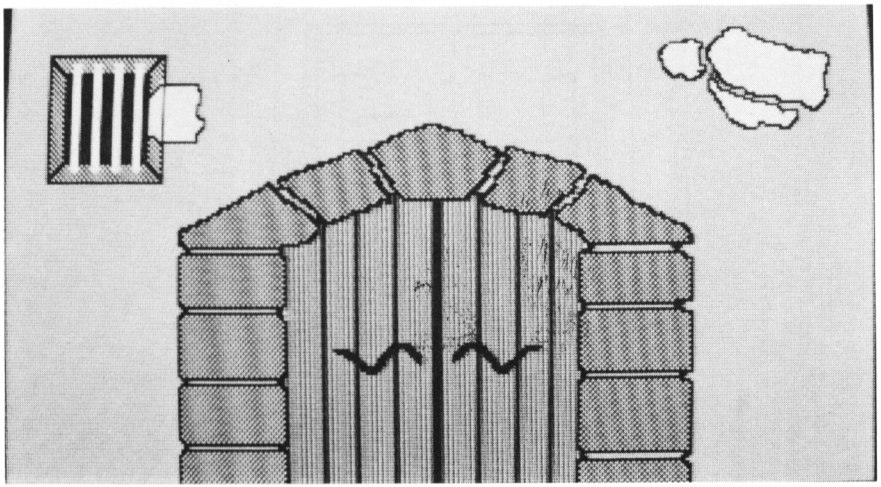
Devant la librairie



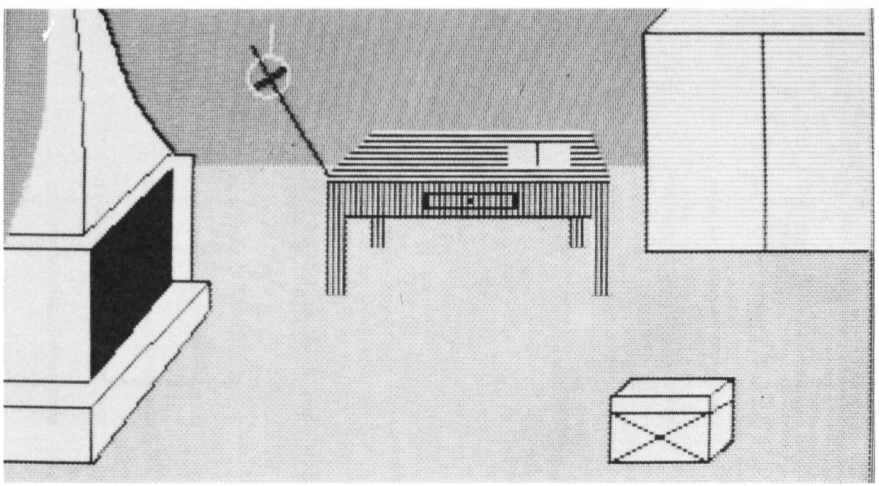
Devant le secrétariat



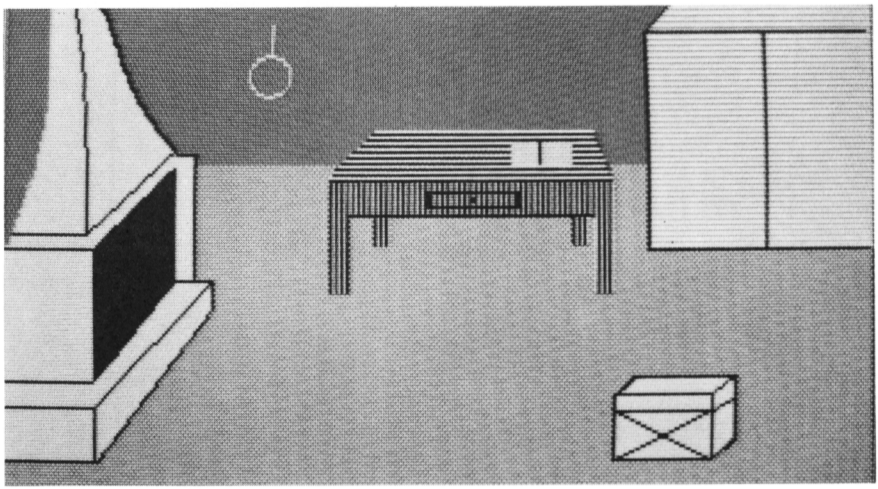
Devant la cour de Versailles



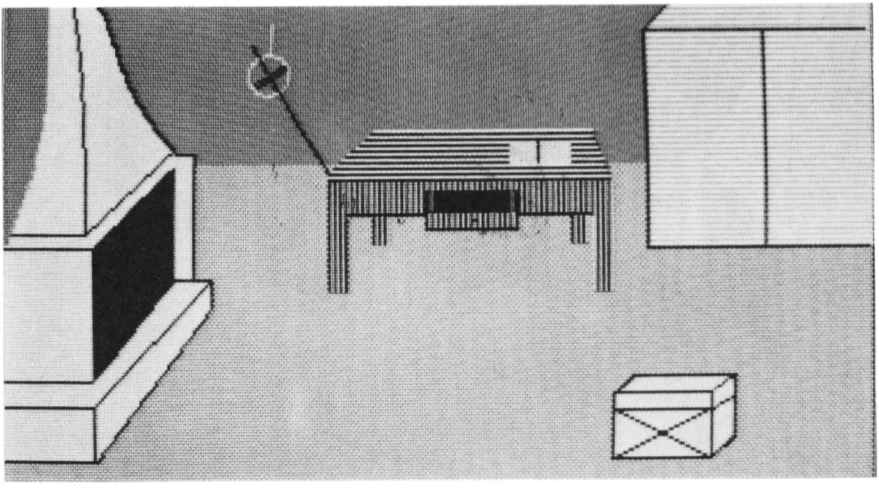
Devant la cour de Buckingham



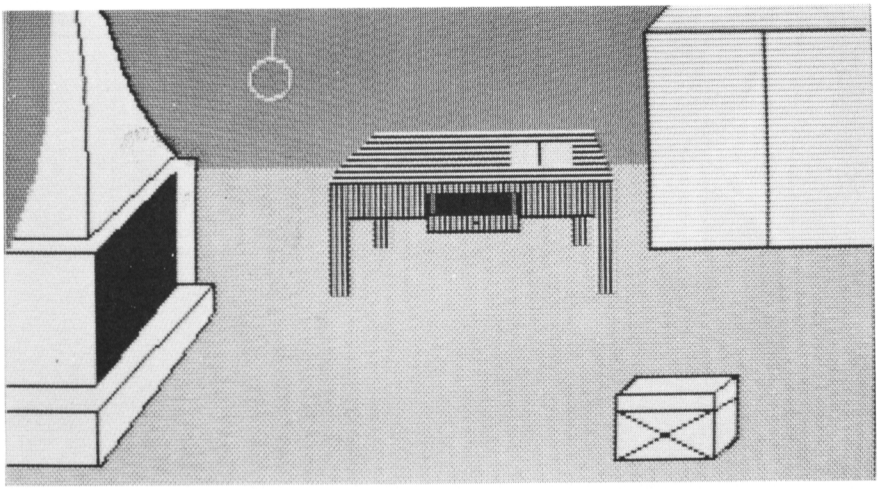
Dans la chambre de d'Artagnan



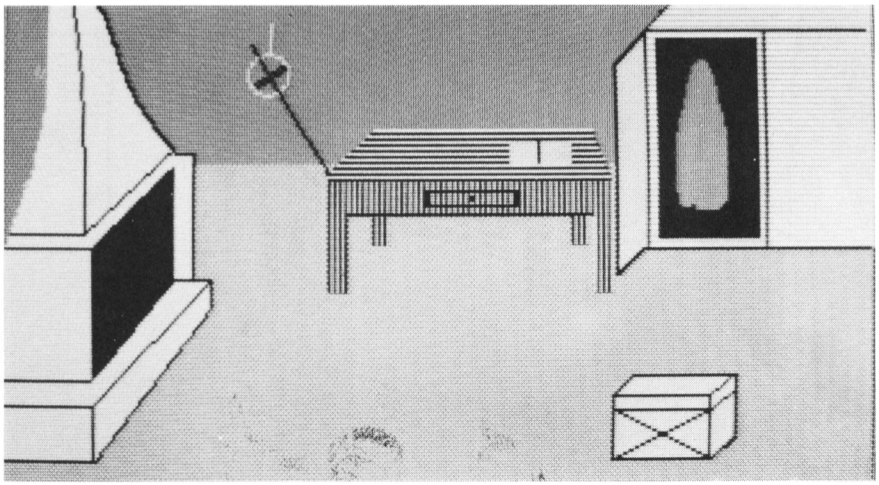
Chambre de d'Artagnan sans épée



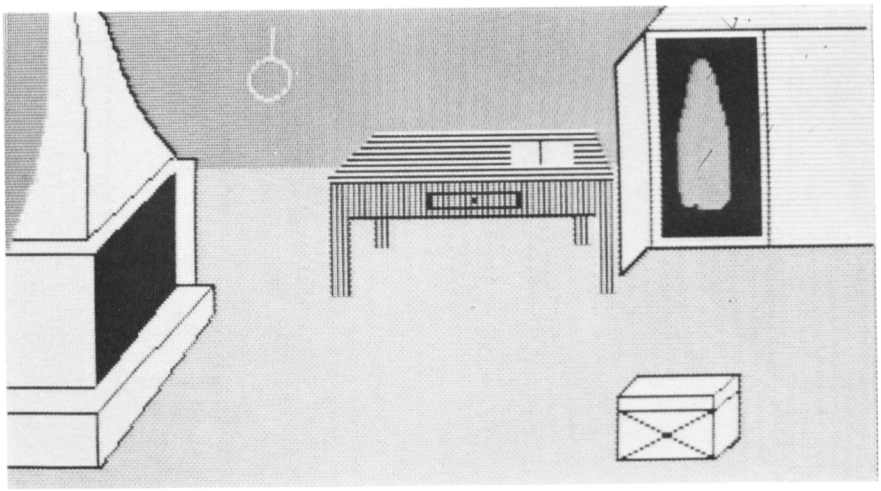
Chambre de d'Artagnan tiroir ouvert



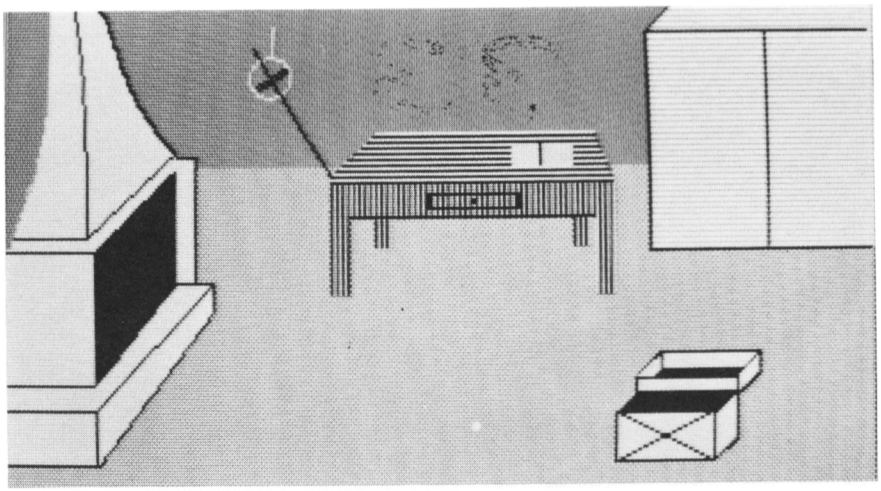
Chambre de d'Artagnan sans épée, tiroir ouvert



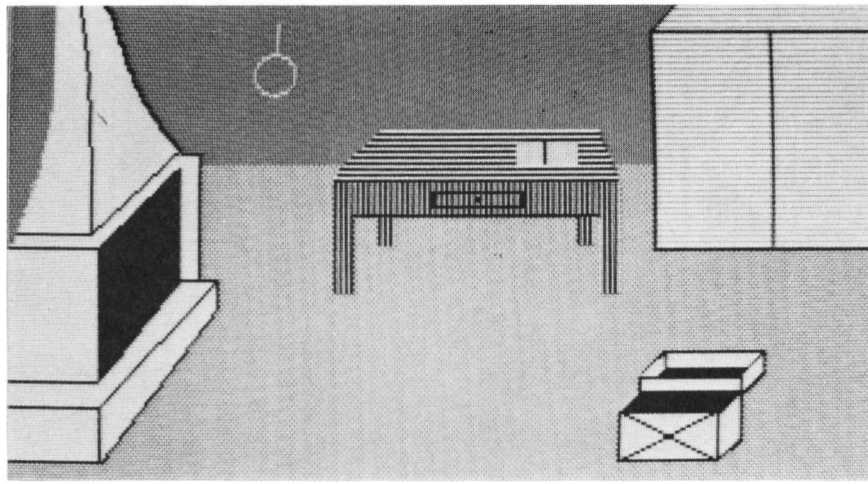
Chambre de d'Artagnan armoire ouverte



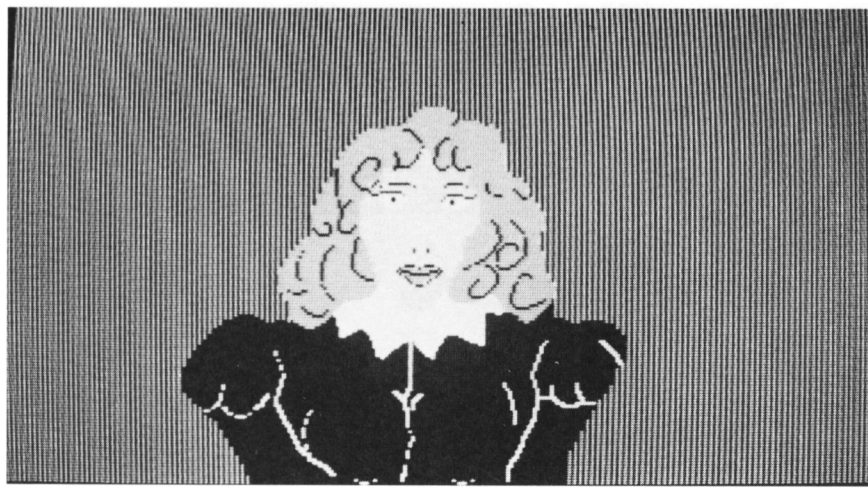
Chambre de d'Artagnan sans épée, armoire ouverte



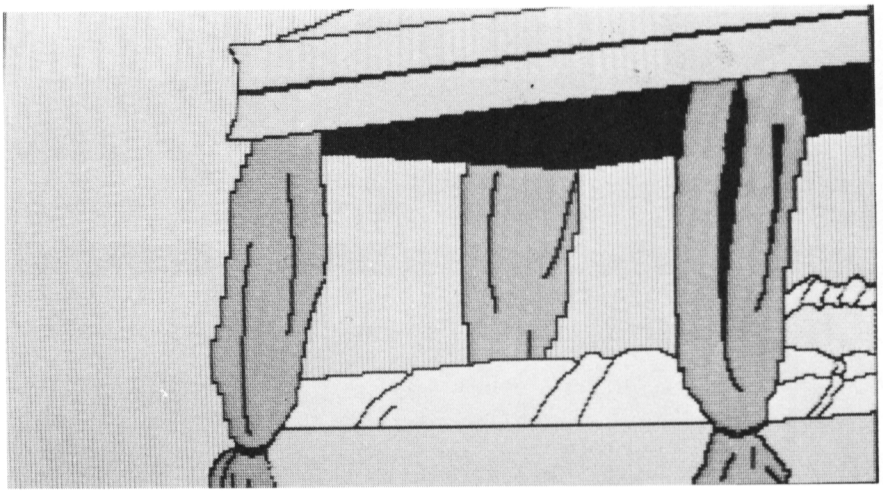
Chambre de d'Artagnan coffre ouvert



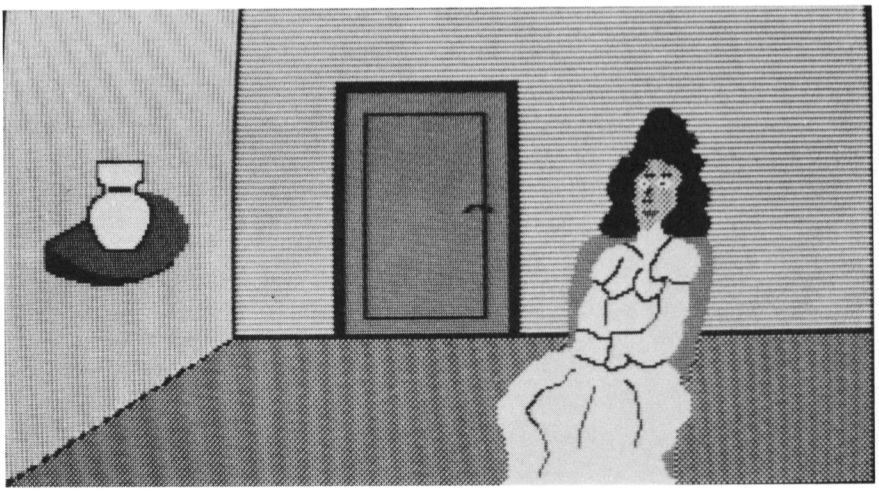
Chambre de d'Artagnan sans épée, coffre ouvert



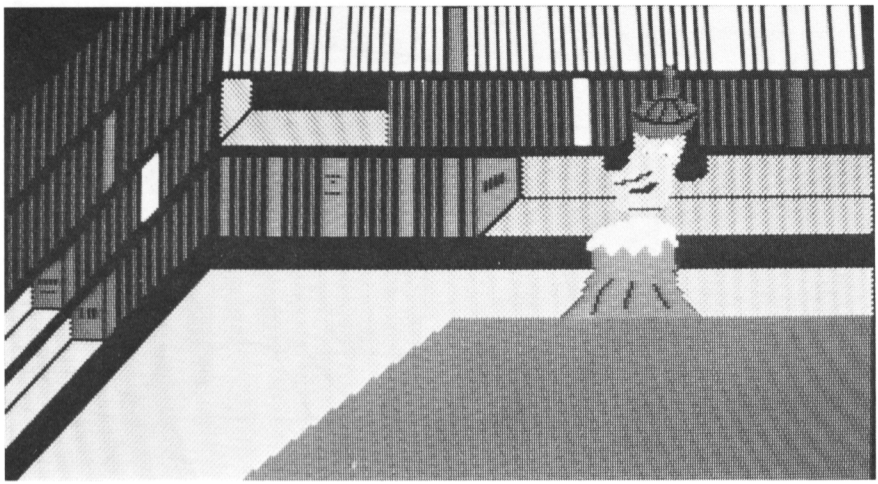
Dans la maison de la veuve



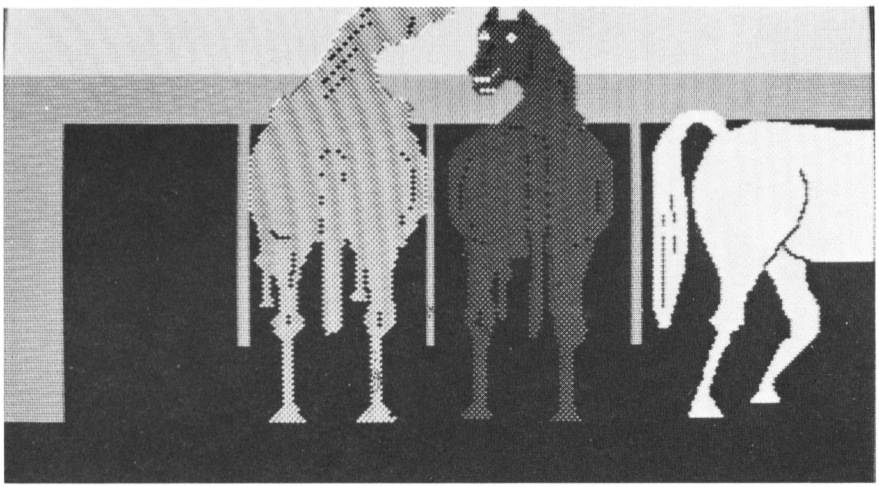
Chambre de la maison de la veuve



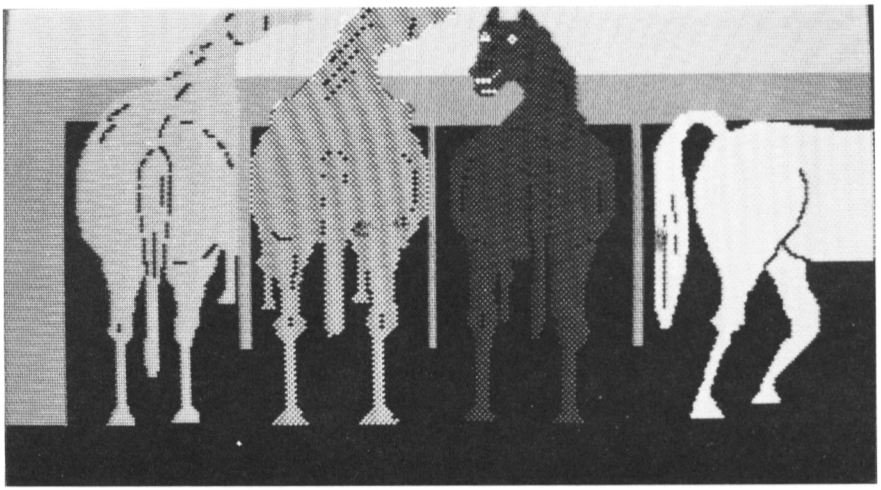
Le boudoir de la reine



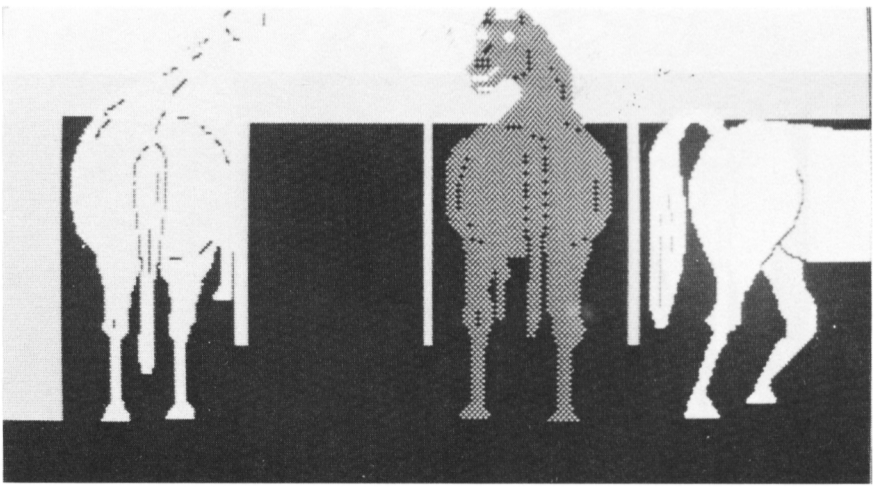
Dans la bibliothèque du duc



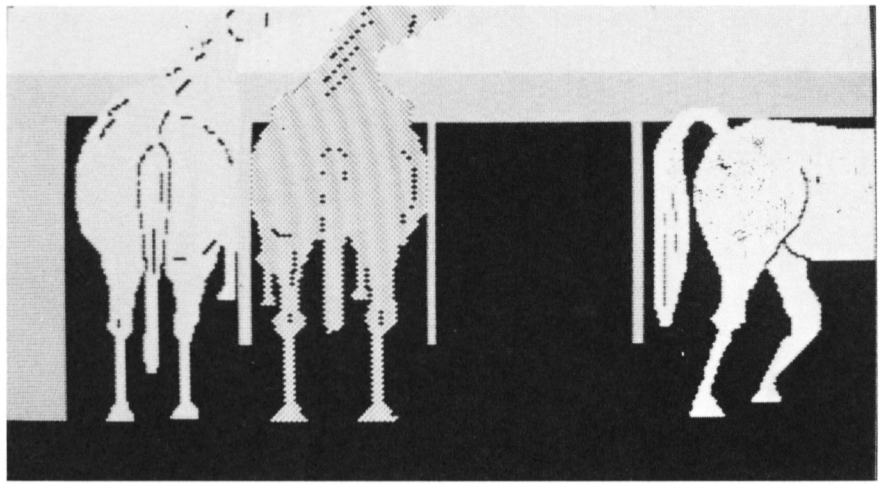
Dans les écuries



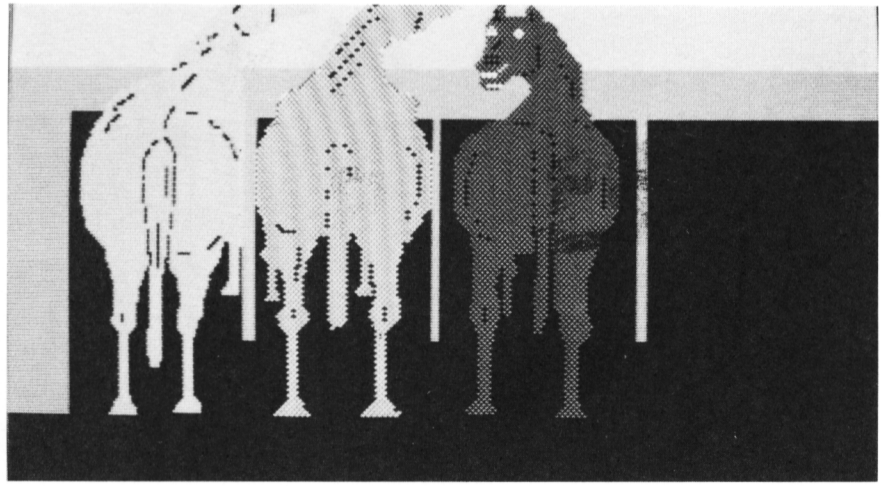
Ecuries sans Bonami



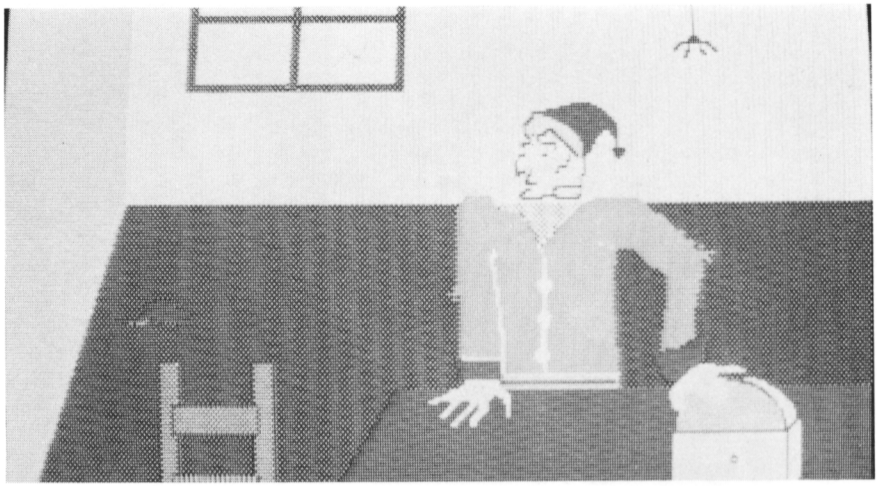
Ecuries sans Flambeau



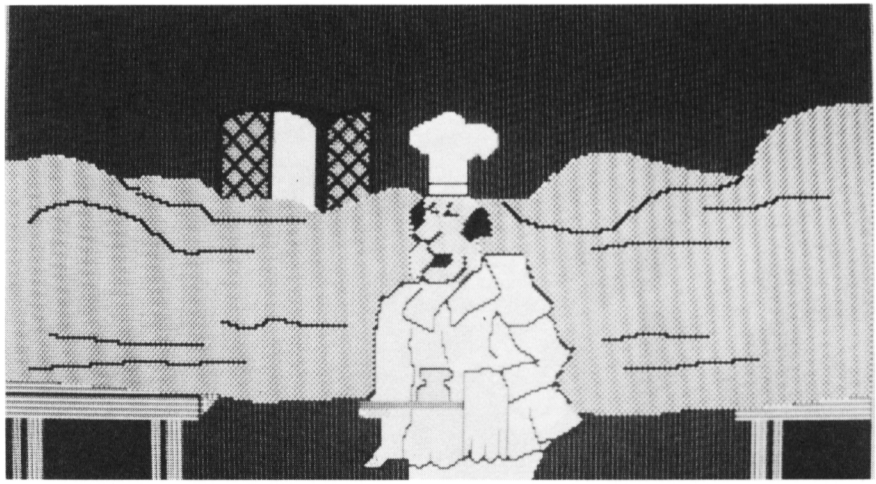
Ecuries sans Laflèche



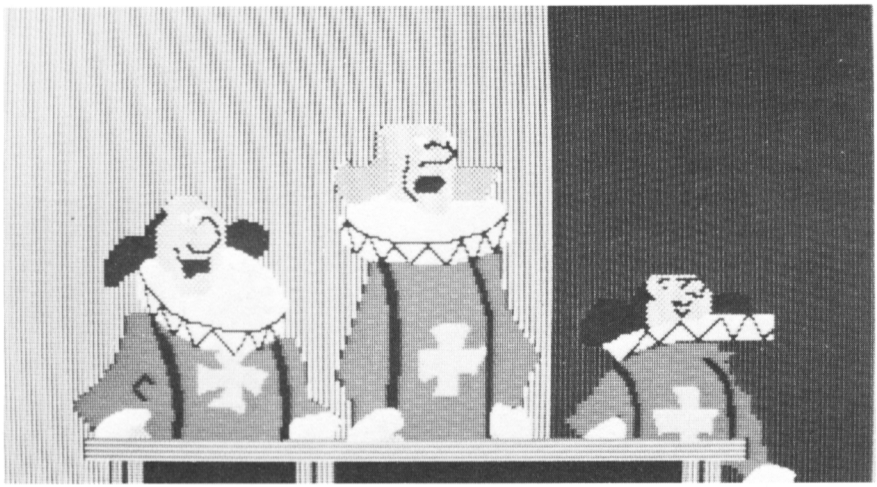
Ecuries sans Mirandole



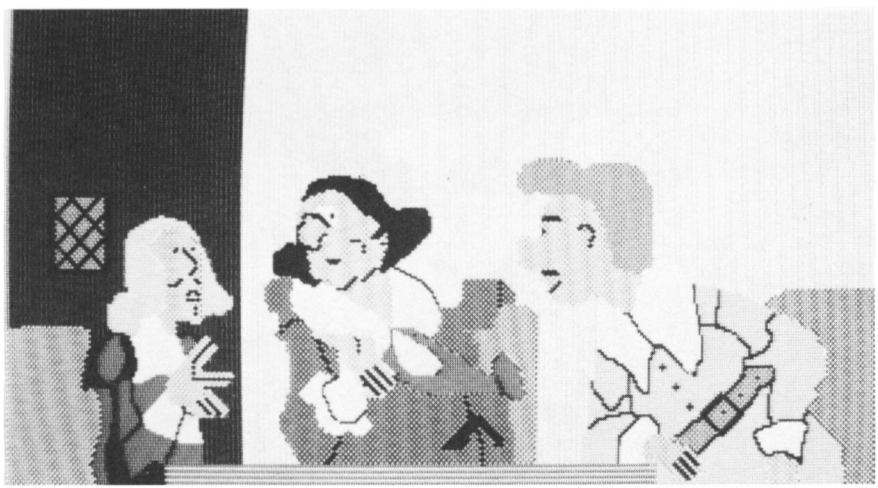
Chez l'usurier



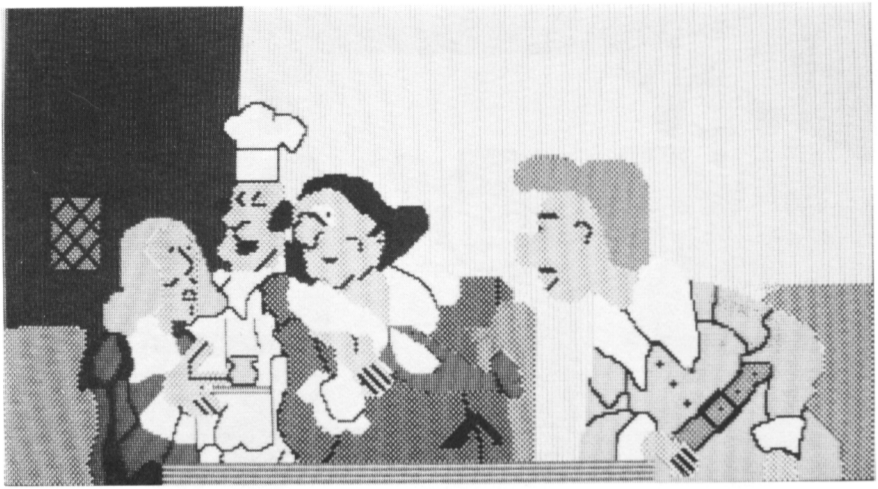
Dans l'estaminet



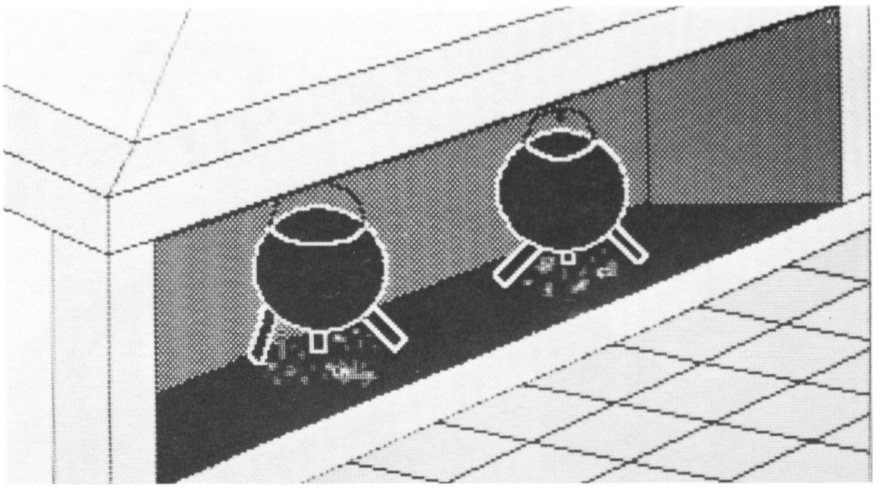
Les gardes du cardinal dans l'estaminet



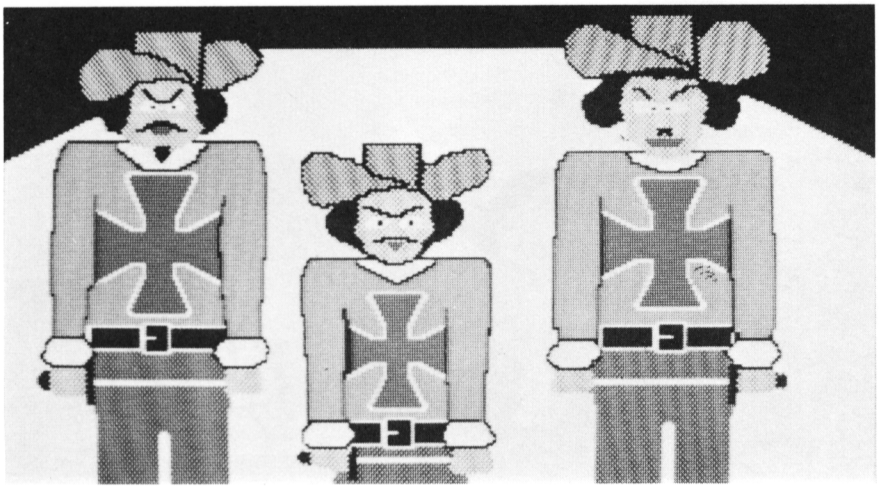
Athos, Porthos et Aramis dans l'estaminet



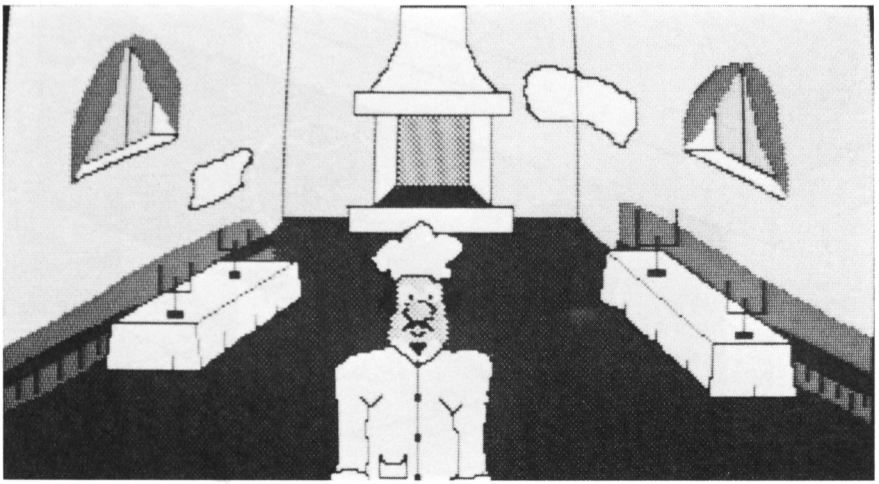
Athos, Porthos, Aramis et le patron dans l'estaminet



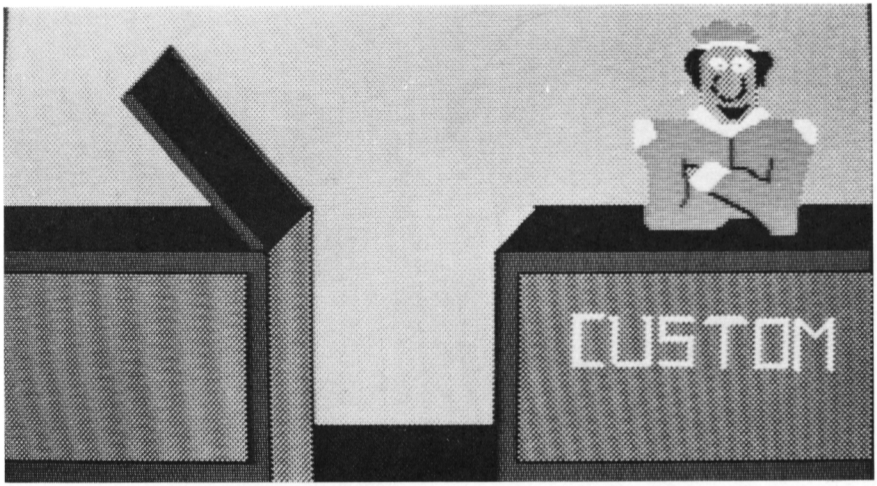
Dans les cuisines



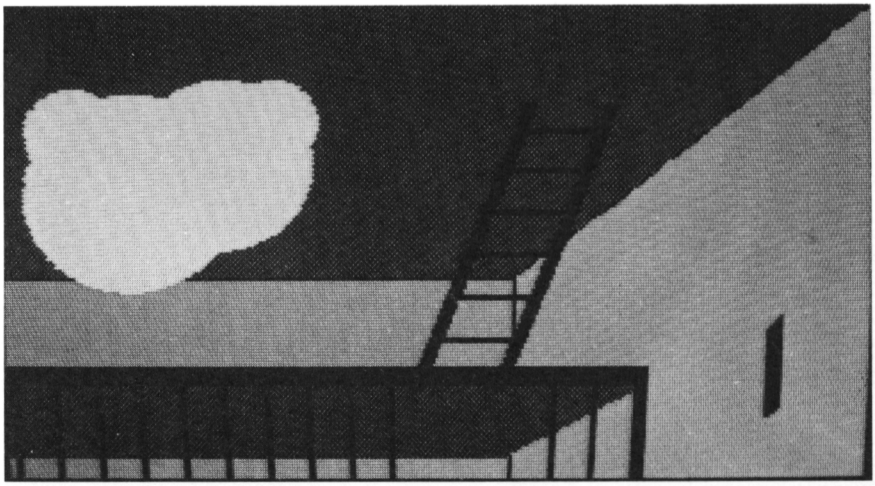
Dans la salle des gardes



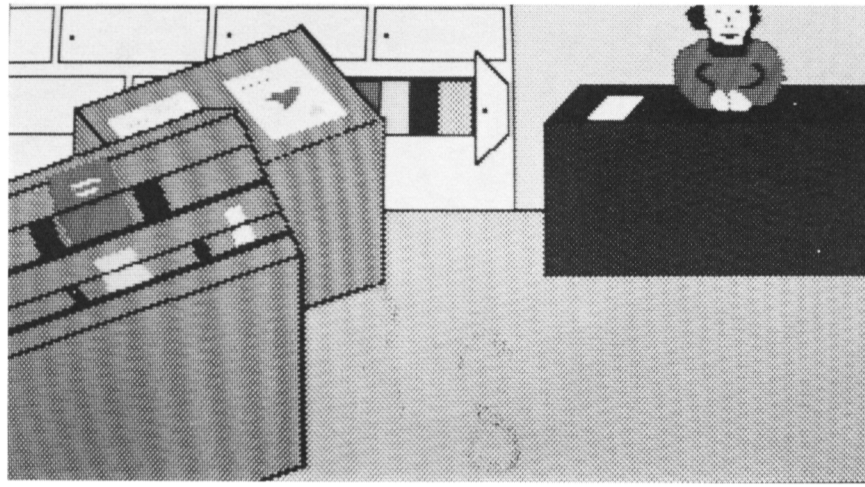
Dans le restaurant



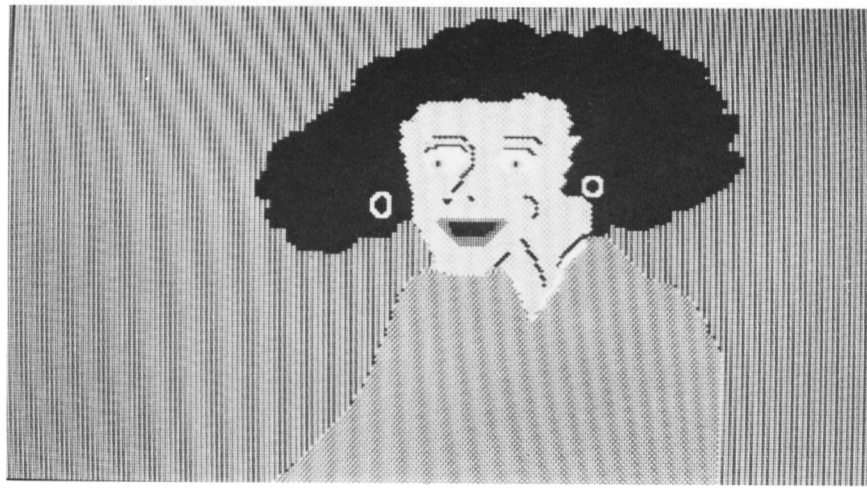
Dans les douanes



Dans la grange



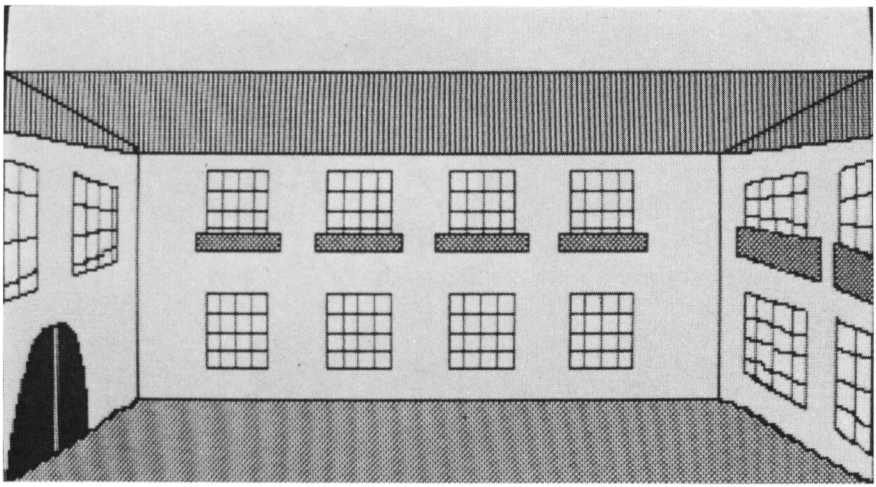
Dans la librairie



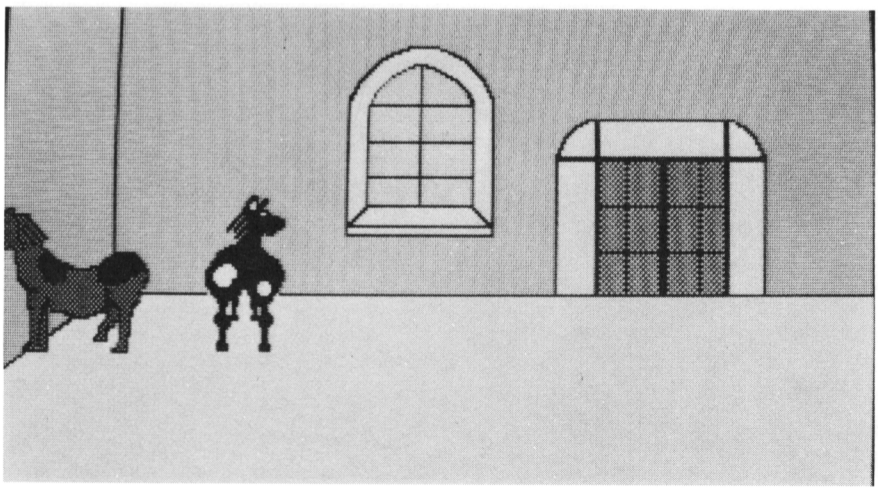
Dans le secrétariat du duc

La secrétaire de Buckingham souriante

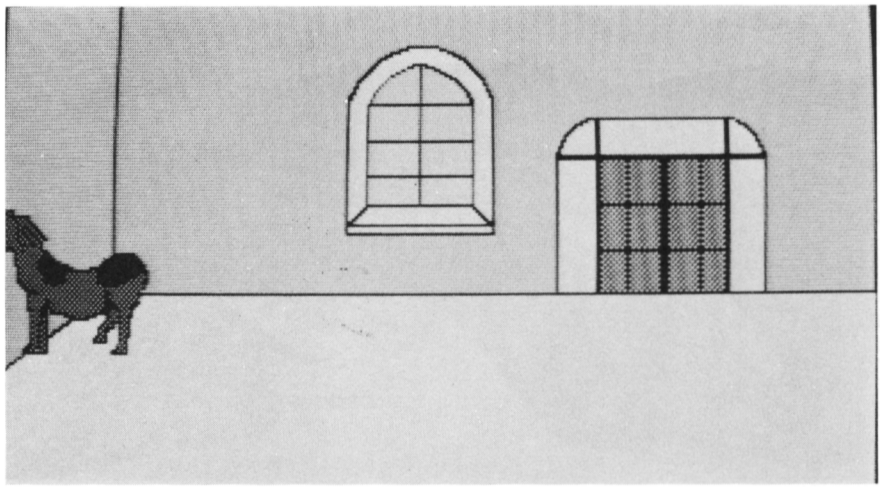
La secrétaire de Buckingham déçue



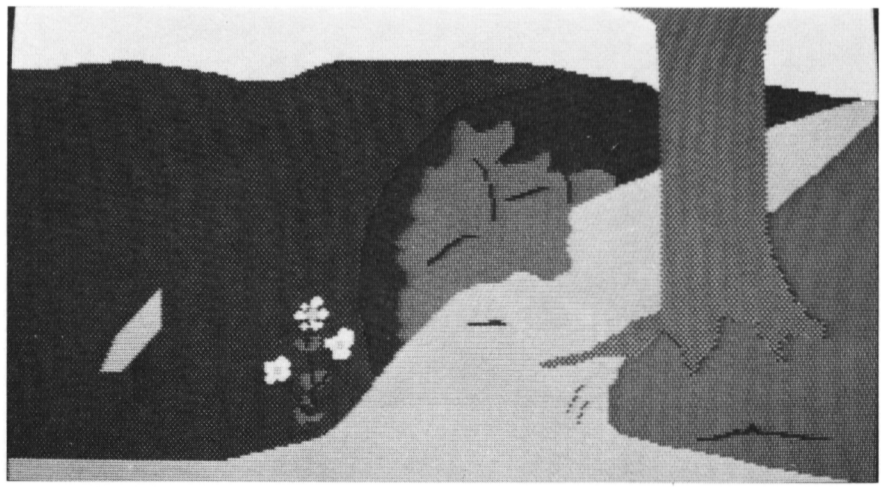
Dans la cour de Versailles



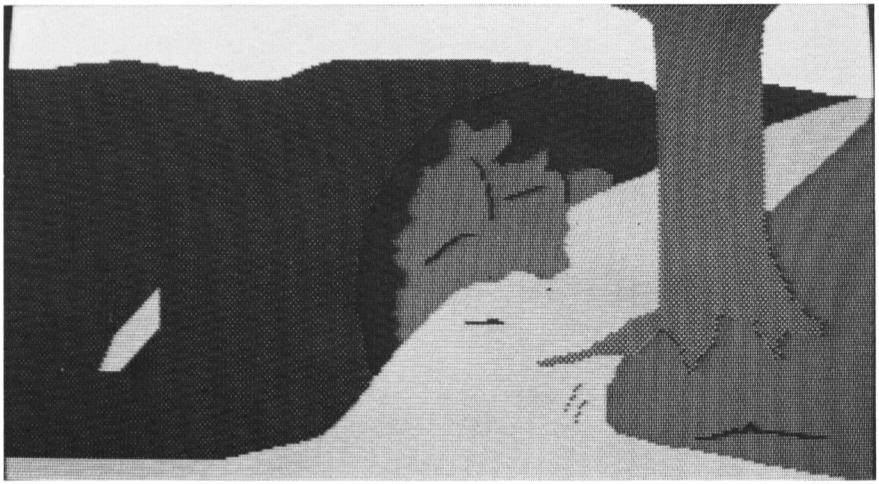
Dans la cour de Buckingham



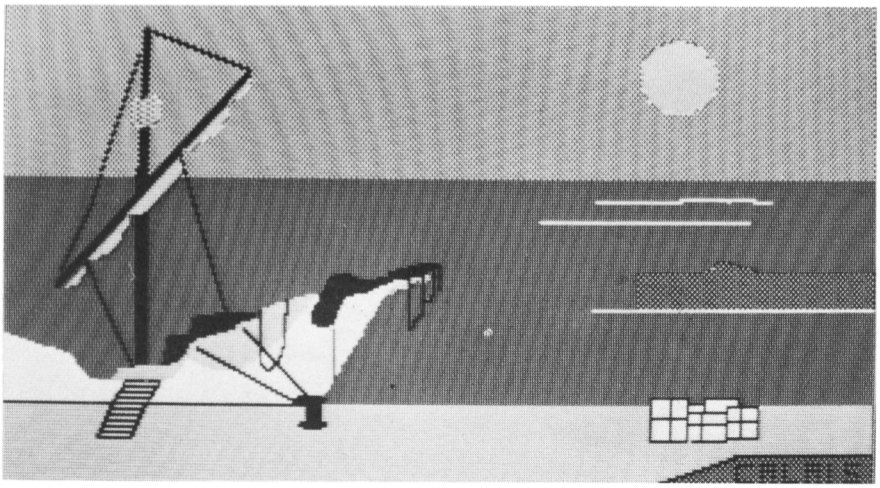
La cour de Buckingham après changement de cheval



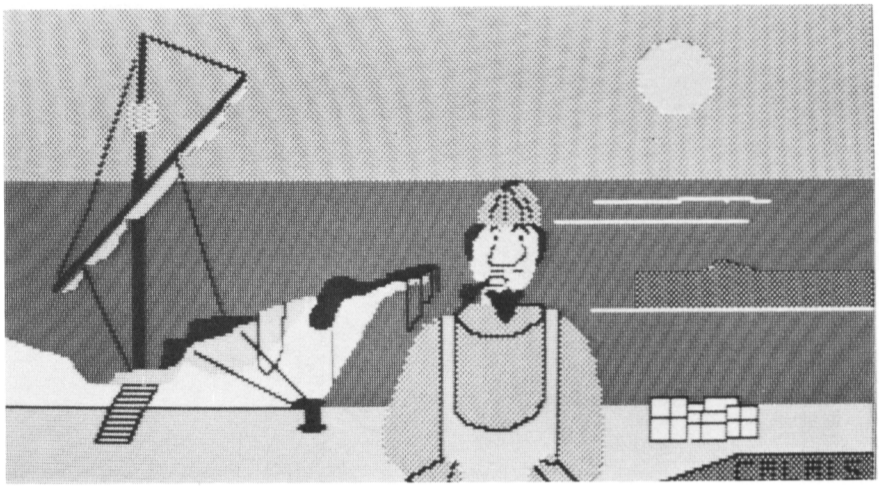
La campagne française



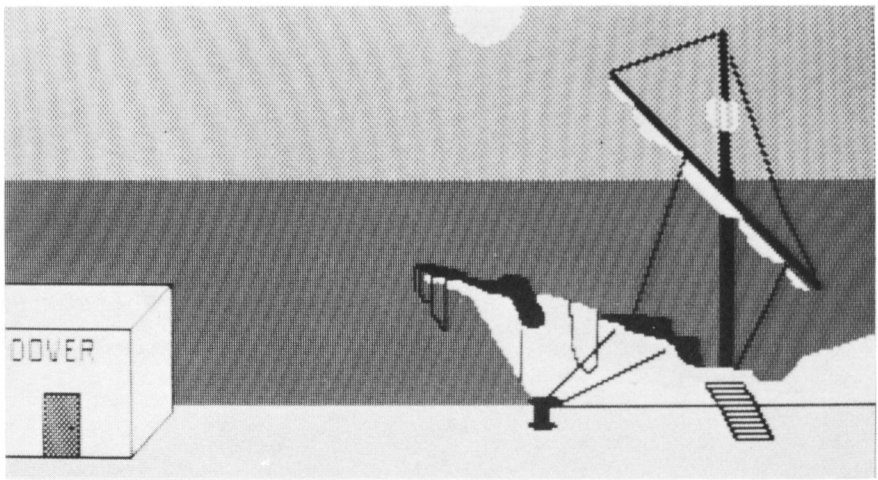
La campagne française sans fleurs



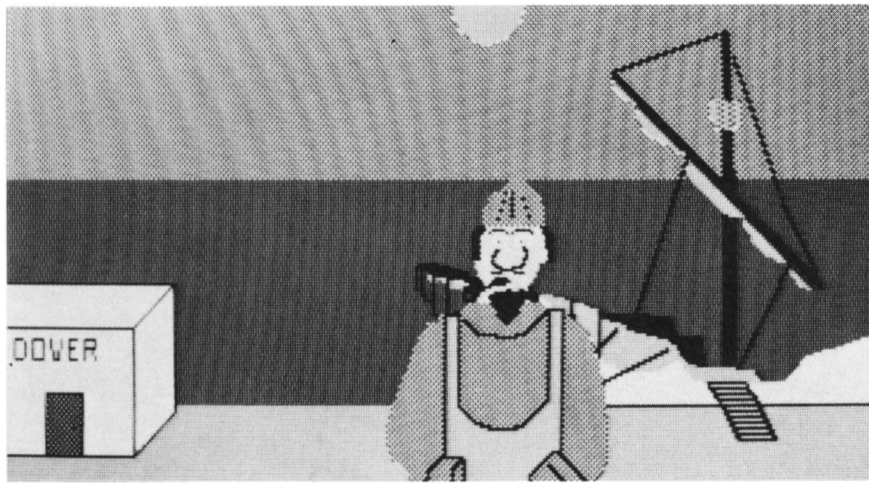
Le port de Calais



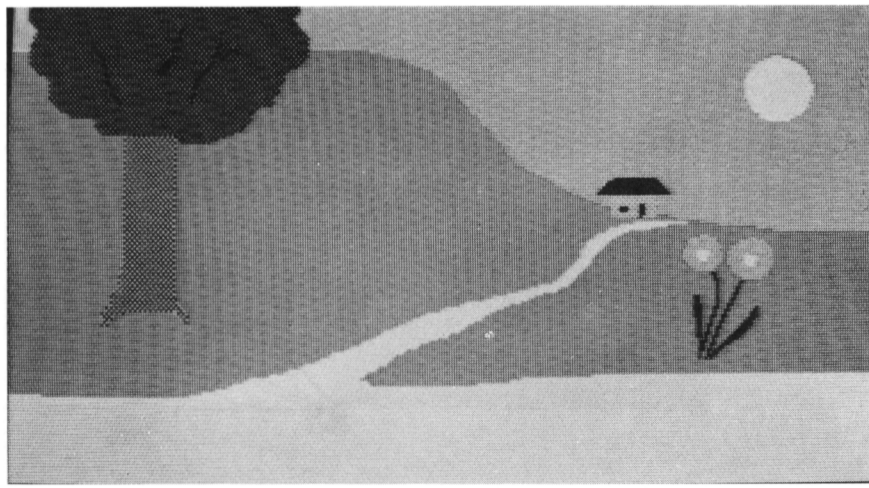
Le port de Calais avec le capitaine



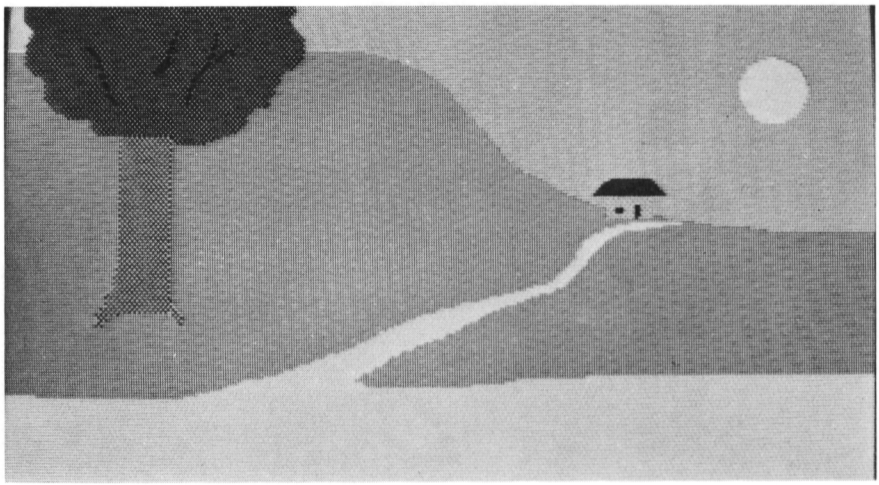
Le port de Douvres



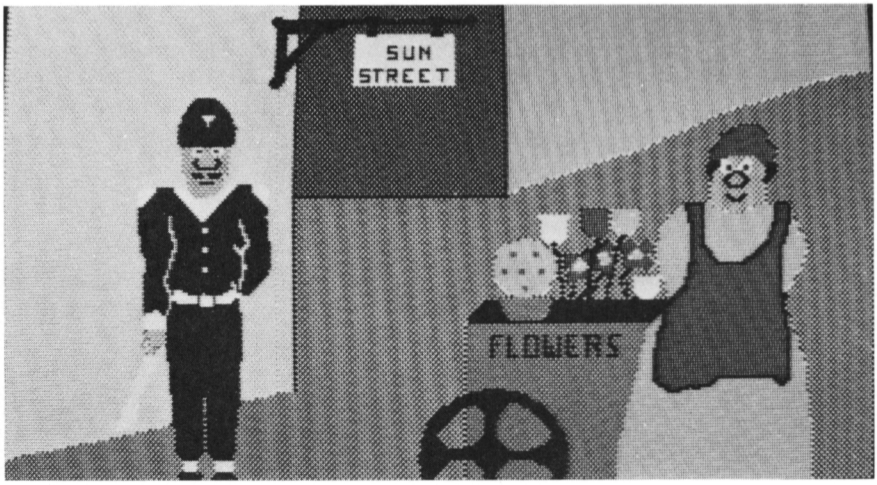
Le port de Douvres avec le capitaine



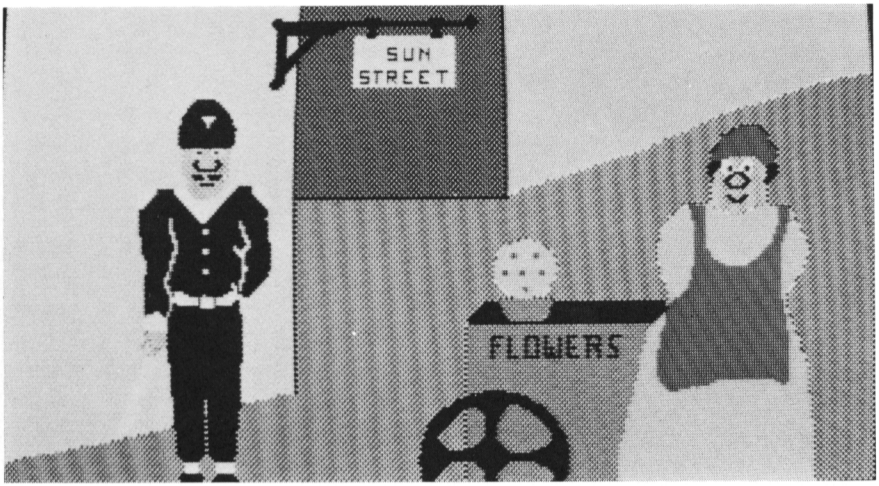
Le champ en Angleterre



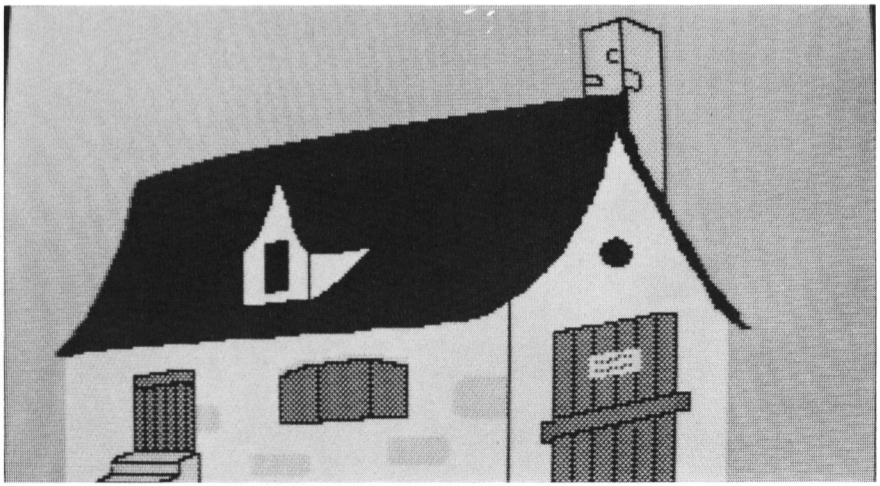
Le champ en Angleterre sans fleurs



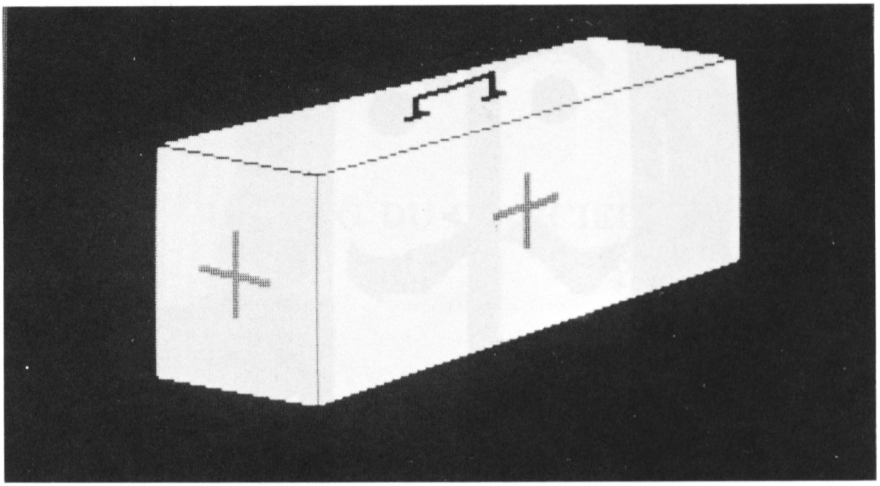
Rue de Londres



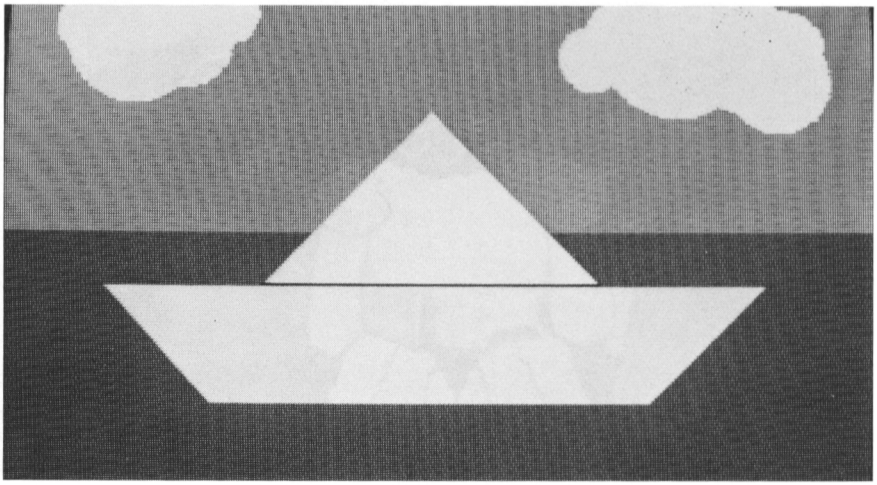
La rue de Londres sans fleurs



Le relais de la Jument bleue



Les soins lorsque l'on est blessé



La mer

Image si l'on a perdu

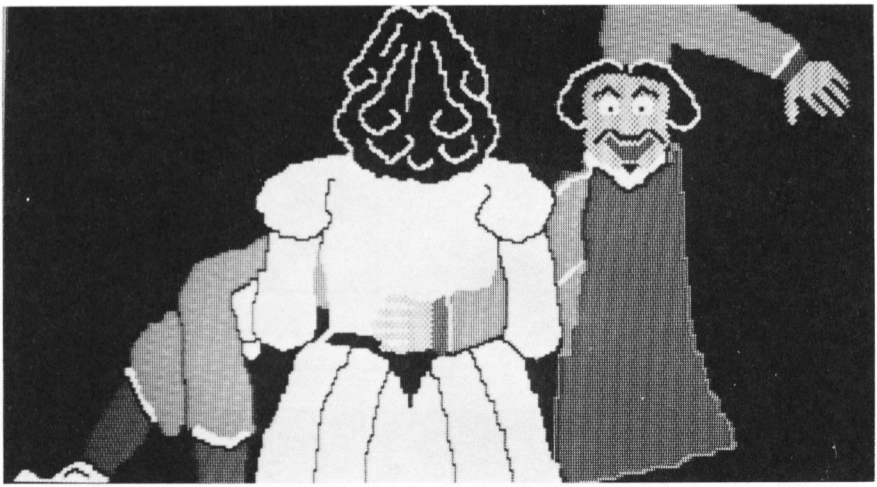


Image si l'on a gagné

C. LISTING DU LOGICIEL CREIMAGE VERSION CPC 464

```
5 ON ERROR GOTO 62000
10 MEMORY 9724:MODE 1:DATA 33,0,0,205,5,
188,201,0,62,,33,,,17,,,205,68,188,201,0
,0,33,0,192,1,91,102,22,1,94,35,125,238,
255,194,42,102,124,238,255,202,66,102,12
3,190,194,57,102,20,202,54,102,195,29,10
2,22,255,0,3,123,2,122,3,2,195
20 DATA 26,102,3,2,122,3,2,237,67,77,102
,201,0,,,0,1,0,64,17,,,33,,,237,176,201,
0,33,0,192,1,91,134,3,10,87,3,10,114,35,
61,194,104,166,125,238,255,194,99,166,12
4,238,255,194,99,166,201,0
30 FOR I=26110 TO 26203:READ A:POKE I,A:
NEXT:FOR I=42588 TO 42619:READ A:POKE I,
A:NEXT:SPEED KEY 10,1:E$="■     "
40 B=1:I0=1:I1=24:I2=8:I3=16:P=0:BORDER
1:GOSUB 14050:LOCATE 12,1:PRINT "CREATIO
N D'IMAGES":LOCATE 15,12:PRINT "EXPLICAT
IONS":LOCATE 17,20:PRINT "TAPEZ <E>":WHI
LE INKEY$="":WEND:CLS
1000 GOSUB 14050:BORDER B:INK 0,I0:INK 1
,I1:INK 2,I2:INK 3,I3:PLOT X,Y,P
1100 CALL 26110:A$="":WHILE A$="":A$=INK
EY$:WEND:A=ASC(UPPER$(A$)):IF A>47 AND A
<52 THEN P=A-48
1110 X=X-2*((A=243)-(A=242)):IF X<0 THEN
 X=0
1120 IF X>638 THEN X=638
1130 Y=Y-2*((A=240)-(A=241)):IF Y<0 THEN
 Y=0
1140 IF Y>398 THEN Y=398
1150 IF A<65 OR A>90 THEN 1000
1160 ON A-64 GOTO 28000,12000,23000,1600
0,29000,24000,1100,1100,14000,25000,1100
,19000,15000,26000,1100,13000,1100,21000
,18000,27000,1100,1100,1100,22000,17000,
20000
12000 GOSUB 60030:LOCATE 1,24:PRINT "BOR
DURE = ";
```

■ = Taper control G.

```
12010 LOCATE 11,24:INPUT "",B:IF B<0 OR
B>26 THEN LOCATE 11,24:PRINT E$:GOTO 120
10:ELSE GOSUB 60040:GOTO 1000
13000 GOSUB 60030:LOCATE 1,24:PRINT "POS
ITION : X= ";XPOS/2;" Y= ";YPOS/2
13010 WHILE INKEY$="":WEND:GOSUB 60040:G
OTO 1100
14000 GOSUB 60030:LOCATE 16,22:PRINT "PA
LETTE :":LOCATE 1,24:PRINT "I0=     I1=
    I2=     I3=     (0-26)";
14010 LOCATE 4,24:INPUT "",I0:IF I0<0 OR
 I0>26 THEN LOCATE 4,24:PRINT E$:GOTO 14
010
14020 LOCATE 12,24:INPUT "",I1:IF I1<0 O
R I1>26 THEN LOCATE 12,24:PRINT E$:GOTO
14020
14030 LOCATE 20,24:INPUT "",I2:IF I2<0 O
R I2>26 THEN LOCATE 20,24:PRINT E$:GOTO
14030
14040 LOCATE 28,24:INPUT "",I3:IF I3<0 O
R I3>26 THEN LOCATE 28,24:PRINT E$:GOTO
14040:ELSE GOSUB 60040:GOTO 1000
14050 POKE 65530,I0:POKE 65531,I1:POKE 6
5532,I2:POKE 65533,I3:POKE 65529,B:RETUR
N
15000 GOSUB 60030:LOCATE 1,24:PRINT "DEP
LACEMENT":GOSUB 61100:X=2*X1:Y=2*Y1:GOTO
 1000
16000 GOSUB 60030:LOCATE 18,25:PRINT "CO
ORDONNEES ABSOLUES":GOSUB 61000:DRAW 2*X
1,2*Y1,P:GOTO 1100
17000 GOSUB 60030:LOCATE 18,25:PRINT "CO
ORDONNEES RELATIVES":GOSUB 61000:DRAWR 2
*X1,2*Y1,P:GOTO 1100
18000 GOSUB 60000:GOSUB 60030:LOCATE 1,2
3:PRINT "SAUVEGARDE NORMALE (1) COMPRESS
EE (2) :";:GOSUB 19100:INPUT "NOM :",B$:
GOSUB 60040:IF A=50 THEN CALL 26132:C=PE
EK(26189)+256*PEEK(26190)-26203:ELSE GOS
UB 60050:SAVE B$,B,26236,16384:GOTO 1803
0
```

```
18010 LOCATE 1,24:PRINT C;:IF C>8192 THE
N PRINT " IMAGE > 8K":WHILE INKEY$="":WE
ND:GOSUB 60040:GOTO 1100
18020 FOR I=0 TO 500:NEXT:POKE 26194,32:
POKE 26196,92:POKE 26197,134:POKE 26199,
92:POKE 26200,102:CALL 26192:POKE 26194,
64:GOSUB 60040:GOSUB 60050:SAVE B$,B,343
96,C:GOSUB 60000
18030 GOSUB 60060:GOTO 1100
19000 GOSUB 60030:LOCATE 1,23:PRINT "CHA
RGEMENT NORMAL (1) DECOMPRESSE (2) :";:G
OSUB 19100:INPUT "NOM :",B$:GOSUB 60050:
LOAD B$:IF A=50 THEN CALL 42589:GOSUB 60
000:ELSE GOSUB 60010
19010 GOSUB 60060:I0=PEEK(65530):I1=PEEK
(65531):I2=PEEK(65532):I3=PEEK(65533):B=
PEEK(65529):GOTO 1000
19100 A$=INKEY$:IF A$="" THEN 19100:ELSE
 A=ASC(A$):IF A<49 OR A>50 THEN GOSUB 60
040:GOTO 1100:ELSE PRINT A$;:RETURN
20000 GOSUB 60030:LOCATE 17,22:PRINT "ZO
NE":LOCATE 1,24:PRINT "RECTANGLE (1) TRI
ANGLE (2) CERCLE (3)   GRAND RECTANGLE (
4)"
20010 A$=INKEY$:IF A$="" THEN 20010 ELSE
 A=ASC(A$):GOSUB 60040:IF A<48 OR A>52 T
HEN 1100:ELSE ON A-48 GOTO 20100,20200,2
0300,20400
20100 GOSUB 60030:LOCATE 15,22:PRINT "RE
CTANGLE":LOCATE 1,24:PRINT "UNIFORME
  (1) RAYE HORIZONTAL    (2)";:LOCATE 1,
25:PRINT "RAYE VERTICAL (3) COULEURS ALT
ERNEES (4)";
20110 A*="":WHILE A$="":A$=INKEY$:WEND:A
=ASC(A$):GOSUB 60040:IF A<48 OR A>52 THE
N 1100:ELSE LOCATE 1,24:PRINT "COULEUR 1
 =    COULEUR 2 =":LOCATE 1,25:PRINT "X1
=       Y1=       X3=       Y3=";
20130 LOCATE 13,24:INPUT "",P1:IF P1<0 O
R P1>3 THEN LOCATE 13,24: PRINT E$:GOTO
20130
```

```
20140 LOCATE 28,24:INPUT "",P2:IF P2<0 O
R P2>3 THEN LOCATE 28,24:PRINT E$:GOTO 2
0140
20150 LOCATE 5,25:INPUT "",X1:LOCATE 15,
25:INPUT "",Y1:LOCATE 25,25:INPUT "",X3:
LOCATE 35,25:INPUT "",Y3:GOSUB 60040:ON
A-48 GOTO 20160,20170,20180,20190
20160 FOR I=Y1 TO Y3:MOVE 2*X1,2*I:DRAW
2*X3,2*I,P1:NEXT:GOTO 1100
20170 FOR I=Y1 TO Y3 STEP 2:MOVE 2*X1,2*
I:DRAW 2*X3,2*I,P1:MOVE 2*X1,2*(I+1):DRA
W 2*X3,2*(I+1),P2:NEXT:GOTO 1100
20180 FOR I=X1 TO X3 STEP 2:MOVE 2*I,2*Y
1:DRAW 2*I,2*Y3,P1:MOVE 2*(I+1),2*Y1:DRA
W 2*(I+1),2*Y3,P2::NEXT:GOTO 1100
20190 FOR I=Y1 TO Y3:FOR K=X1 TO X3 STEP
 2:PLOT 2*K,2*I,P1:PLOT 2*(K+1),2*I,P2:N
EXT:P=P2:P2=P1:P1=P:NEXT:GOTO 1100
20200 GOSUB 61300:L=SQR((X3-X2)^2+(Y3-Y2
)^2):DX=(X3-X2)/2/L:DY=(Y3-Y2)/2/L:A=4*I
NT(L):MOVE 2*X1,2*Y1:I=2*X2:K=2*Y2:DRAW
I,K,P:FOR J=1 TO A:MOVE 2*X1,2*Y1:I=I+DX
:K=K+DY:DRAW I,K,P:NEXT:GOTO 1100
20300 GOSUB 61200:FOR I=-PI/2 TO PI/2 ST
EP 1/R:K=2.2*R*COS(I):ORIGIN 2*XC-K,2*YC
+2*R*SIN(I):DRAW 2*K,0,P:NEXT:ORIGIN 0,0
:GOTO 1100
20400 GOSUB 60040:LOCATE 1,24:PRINT "COU
LEUR =         X1=         Y1=":LOCATE 18,
25:PRINT "X2=         Y2="
20410 LOCATE 11,24:INPUT "",P1:IF P1<0 O
R P1>255 THEN LOCATE 11,24:PRINT E$:GOTO
 20410
20420 LOCATE 22,24:INPUT "",X1:IF X1<0 O
R X1>39 THEN LOCATE 22,24:PRINT E$:GOTO
20420
20430 LOCATE 33,24:INPUT ""
,Y1:IF Y1<0 OR Y1>24 THEN LOCATE 33,24:P
RINT E$:GOTO 20430
20440 LOCATE 22,25:INPUT "",X2:IF X2<0 O
R X2>39 THEN LOCATE 22,25:PRINT E$:GOTO
20440
```

```
20450 LOCATE 33,25:INPUT "",Y2:IF Y2<0 O
R Y2>24 THEN LOCATE 33,25:PRINT E$:GOTO
20450
20460 GOSUB 60040:POKE 26119,P1:POKE 261
21,Y1 :POKE 26122,X1 :POKE 26124,Y2 :POK
E 26125,X2:CALL 26117:GOTO 1100
21000 GOSUB 60010:GOTO 1100
22000 GOSUB 60030:CLS:PRINT "0 NOIR
 9  VERT        18 VERT VIF    1 BLEU
 10 TURQUOISE   19 VERT MARIN 2 BLEU VIF
 11 BLEU CIEL   20 TURQUOI VIF3 ROUGE
 12 JAUNE       21 VERT CITRON4 MAGENTA
 13 BLANC       22 VERT PASTEL";
22010 PRINT "5 MAUVE      14 BLEU PAST  2
3 TURQU PAST 6 ROUGE VIF 15 ORANGE      2
4 JAUNE VIF  7 POURPRE   16 ROSE        2
5 JAUNE PAST 8 MAGEN VIF 17 MAGEN PAST 2
6 BLANC BRILL"
22020 PRINT "INITIAL 1-24-16-8":PRINT:PR
INT "ACTUEL ";I0;I1;I2;I3;" BORDURE :";B
22030 WHILE INKEY$="":WEND:GOSUB 60040:G
OTO 1100
23000 GOSUB 60000:GOTO 1100
24000 GOSUB 60030:LOCATE 16,22:PRINT "FI
GURE":LOCATE 1,24:PRINT     "RECTANGLE (1
) TRIANGLE (2) CERCLE (3)"
24010 A$=INKEY$:IF A$="" THEN 24010 ELSE
 A=ASC(A$)
24020 GOSUB 60040:IF A<48 OR A>51 THEN 1
100:ELSE ON A-48 GOTO 24100,24200,24300
24100 LOCATE 1,24:PRINT "COULEUR =
 X1 =        Y1 =":LOCATE 17,25:PRINT "X3
 =        Y3 ="
24110 LOCATE 11,24:INPUT "",P:IF P<0 OR
P>3 THEN LOCATE 11,24:PRINT E$:GOTO 2411
0
24120 LOCATE 22,24:INPUT "",X1:LOCATE 33
,24:INPUT "",Y1:LOCATE 22,25:INPUT "",X3
:LOCATE 33,25:INPUT "",Y3:GOSUB 60040
24130 MOVE 2*X1,2*Y1:DRAW 2*X3,2*Y1,P:DR
AW 2*X3,2*Y3:DRAW 2*X1,2*Y3:DRAW 2*X1,2*
Y1:GOTO 1100
```

```
24200 GOSUB 61300
24210 MOVE 2*X1,2*Y1:DRAW 2*X2,2*Y2,P:DR
AW 2*X3,2*Y3:DRAW 2*X1,2*Y1:GOTO 1100
24300 GOSUB 61200:ORIGIN 2*(XC+1.1*R),2*
YC:FOR I=0 TO 2*PI+(1/R) STEP 1/R:DRAW 2
.2*R*COS(I)-2.2*R,2*R*SIN(I),P:NEXT:ORIG
IN 0,0:GOTO 1100
25000 GOSUB 60030:LOCATE 1,24:INPUT "IMA
GE A SUPERPOSER :",A$:GOSUB 60040:GOSUB
60050:LOAD A$:GOSUB 60060:GOSUB 60020:GO
SUB 60000
25100 FOR K=9726 TO 26109:IF PEEK(K)=0 T
HEN 25160 ELSE A$=BIN$(PEEK(K+39426),8):
B$=BIN$(PEEK(K),8)
25110 A0$=RIGHT$(A$,1):B0$=RIGHT$(B$,1):
A4$=MID$(A$,4,1):B4$=MID$(B$,4,1):IF B0$
<>"0" OR B4$<>"0" THEN A0$=B0$:A4$=B4$
25120 A1$=MID$(A$,7,1):B1$=MID$(B$,7,1):
A5$=MID$(A$,3,1):B5$=MID$(B$,3,1):IF B1$
<>"0" OR B5$<>"0" THEN A1$=B1$:A5$=B5$
25130 A2$=MID$(A$,6,1):B2$=MID$(B$,6,1):
A6$=MID$(A$,2,1):B6$=MID$(B$,2,1):IF B2$
<>"0" OR B6$<>"0" THEN A2$=B2$:A6$=B6$
25140 A3$=MID$(A$,5,1):B3$=MID$(B$,5,1):
A7$=MID$(A$,1,1):B7$=MID$(B$,1,1):IF B3$
<>"0" OR B7$<>"0" THEN A3$=B3$:A7$=B7$
25150 POKE K+39426,128*VAL(A7$)+64*VAL(A
6$)+32*VAL(A5$)+16*VAL(A4$)+8*VAL(A3$)+4
*VAL(A2$)+2*VAL(A1$)+VAL(A0$)
25160 NEXT:GOTO 1100
26000 GOSUB 60030:LOCATE 1,24:PRINT "CON
FIRMER LE NETTOYAGE ECRAN <OUI/NON>"
26010 A$=INKEY$:IF A$="" THEN 26010:ELSE
 IF UPPER$(A$)="O" THEN CLS:GOTO 1000:EL
SE GOSUB 60040:GOTO 1000
27000 GOSUB 60030:CLS:LOCATE 1,1:|DIR
27010 WHILE INKEY$="":WEND:GOSUB 60040:G
OTO 1000
28000 GOSUB 60030:LOCATE 1,23:INPUT "ANN
ULATION DU FICHIER :",B$:PRINT "CONFIRME
R L'ANNULATION DE ";B$:PRINT "<OUI/NON>"
```

```
28010 A$=INKEY$:IF A$="" THEN 28010:ELSE
 IF UPPER$(A$)="O" THEN |ERA,@B$
28020 GOSUB 60040:GOTO 1100
29000 GOSUB 60030:CLS:PRINT "<S> : SAUVE
GARDE IMAGE1":PRINT "<L> : CHARGE IMAGE1
":PRINT "<C> : COPIE IMAGE1 > IMAGE2":PR
INT "<R> : COPIE IMAGE2 > IMAGE1":PRINT
"<N> : NETTOYAGE IMAGE1":PRINT "<T> : CA
TALOGUE":PRINT "<A> : ANNULATION FICHIER
"
29010 PRINT "<P> : POSITION DU POINT":PR
INT "<M> : MOUVEMEMT DU POINT":PRINT "<B
> : COULEUR BORDURE":PRINT "<J> : JUXTAP
OSITION D'IMAGES":PRINT "<I> : COULEUR D
ES ENCRES":PRINT "<X> : CODES DES ENCRES
"
29020 PRINT "<D> : DROITE (COORD ABS)":P
RINT "<Y> : DROITE (COORD RELAT)":PRINT
"<F> : FIGURE":PRINT "<Z> : ZONE":PRINT:
PRINT "<0>,<1>,<2>,<3> : STYLOS":PRINT:P
RINT "FLECHES : DEPLACEMENT"
29100 WHILE INKEY$="":WEND:GOSUB 60040:G
OTO 1000
60000 POKE 26196,92:POKE 26197,102:POKE
26199,0:POKE 26200,192:CALL 26192:RETURN
60010 POKE 26196,0:POKE 26197,192:POKE 2
6199,92:POKE 26200,102:CALL 26192:RETURN
60020 POKE 26196,254:POKE 26197,37:POKE
26199,92:POKE 26200,102:CALL 26192:RETUR
N
60030 POKE 26196,253:POKE 26197,37:POKE
26199,0:POKE 26200,192:CALL 26192:LOCATE
 1,25:PRINT FRE(""):LOCATE 1,25:PRINT "
     ":RETURN
60040 POKE 26196,0:POKE 26197,192:POKE 2
6199,253:POKE 26200,37:CALL 26192:RETURN
60050 POKE 45200,101:POKE 44668,101:RETU
RN
60060 POKE 45200,37:POKE 44668,37:RETURN
61000 LOCATE 1,24:PRINT "ENCRE= "
```

```
61010 LOCATE 8,24:INPUT "",P:IF P<0 OR P
>3 THEN LOCATE 8,24:PRINT E$:GOTO 61010
61100 LOCATE 18,24:PRINT "X =          Y
="
61110 LOCATE 22,24:INPUT "",X1:LOCATE 34
,24:INPUT "",Y1:GOSUB 60040:RETURN
61200 GOSUB 60040:LOCATE 1,24:PRINT "COU
L=     RAYON=       XC=       YC=";
61210 LOCATE 7,24:INPUT "",P:IF P<0 OR P
>3 THEN LOCATE 7,24:PRINT E$:GOTO 61210
61220 LOCATE 18,24:INPUT "",R:LOCATE 27,
24:INPUT "",XC:LOCATE 36,24:INPUT "",YC:
GOSUB 60040:RETURN
61300 GOSUB 60040:LOCATE 1,23:PRINT "COU
LEUR =            X1=        Y1=        X2=
       Y2=        X3=        Y3=";
61310 LOCATE 11,23:INPUT "",P:IF P<0 OR
P>3 THEN LOCATE 11,23:PRINT E$:GOTO 6131
0
61320 LOCATE 25,23:INPUT "",X1:LOCATE 35
,23:INPUT "",Y1:LOCATE 5,24:INPUT "",X2:
LOCATE 15,24:INPUT "",Y2:LOCATE 25,24:IN
PUT "",X3:LOCATE 35,24:INPUT "",Y3:GOSUB
 60040:RETURN
62000 GOSUB 60030:CLS:PRINT "ERREUR BASI
C":WHILE INKEY$="":WEND:GOSUB 60040:RESU
ME 1000
```

D. LISTING DES MODIFICATIONS DE CREIMAGE POUR CPC 6128

```
18000 GOSUB 60000:GOSUB 60030:LOCATE 1,2
3:PRINT "SAUVEGARDE NORMALE (1) COMPRESS
EE (2) :";:GOSUB 19100:INPUT "NOM :",B$:
GOSUB 60040:IF A=50 THEN CALL 26132:C=PE
EK(26189)+256*PEEK(26190)-26203:ELSE MEM
ORY 16384:SAVE B$,B,26236,16384:GOTO 180
30

18020 FOR I=0 TO 500:NEXT:POKE 26194,32:
POKE 26196,92:POKE 26197,134:POKE 26199,
92:POKE 26200,102:CALL 26192:POKE 26194,
64:GOSUB 60040:MEMORY 16384:SAVE B$,B,34
396,C:GOSUB 60000


18030 MEMORY 9724:GOTO 1100

19000 GOSUB 60030:LOCATE 1,23:PRINT "CHA
RGEMENT NORMAL (1) DECOMPRESSE (2) :";:G
OSUB 19100:INPUT "NOM :",B$:MEMORY 16384
:LOAD B$:IF A=50 THEN CALL 42589:GOSUB 6
0000:ELSE GOSUB 60010

19010 MEMORY 9724:I0=PEEK(65530):I1=PEEK
(65531):I2=PEEK(65532):I3=PEEK(65533):B=
PEEK(65529):GOTO 1000

25000 GOSUB 60030:LOCATE 1,25:INPUT "IMA
GE A SUPERPOSER :",A$:GOSUB 60040:MEMORY
 16384:LOAD A$:MEMORY 9724:GOSUB 60020
```

E. LISTING DU PROGRAMME D'AVENTURE DÉBUT. AVENT. FIN.

PROGRAMME DÉBUT

```
1 MODE 2
10 PRINT "> > > > > > > > > > > > > > > > >
  AVERTISSEMENT    < < < < < < < < < < <
< < < <"
20 PRINT "
..................."
30 PRINT:PRINT:PRINT "              LES DE
UX DISQUETTES QUE VOUS VENEZ D'ACQUERRIR
 ONT ETE CONCUES POUR SERVIR DE SUPPORT
AU LIVRE 'AVENTURE SUR AMSTRAD' EDITE CH
EZ SYBEX. VOUS POUVEZ"
35 PRINT "CEPENDANT, SI VOUS NE POSSEDEZ
 PAS CET OUVRAGE, LES UTILISER TOUT A FA
IT NORMA- LEMENT."
40 PRINT:PRINT "              CET OUVRAGE
VOUS APPREND, EN DETAIL, COMMENT CONCEVO
IR, REALISER ET PROGRAMMER VOUS-MEME UN
JEU D'AVENTURE SUR VOTRE CHER AMSTRAD. V
OUS DECOUVRIREZ PEUT-ETRE AUSSI, DES TRU
CS OU ASTUCES QUE VOUS POURREZ UTILISER
DANS D'AUTRES"
45 PRINT "PROGRAMMES."
50 PRINT:PRINT "              LES PETITS F
OUINEURS QUI ONT EU LA CURIOSITE DE REGA
RDER LE CATALO- GUE DE CES DISQUETTES, O
NT PEUT-ETRE ETE SURPRIS D'Y DECOUVRIR L
ES PROGRAMMES"
60 PRINT "'CREI464' ET 'CREI6128'. IL S'
AGIT, TOUT SIMPLEMENT, DE PROGRAMMES DE
CREATION  D'IMAGES PUISSANTS, ET CEPENDA
NT SIMPLE D'EMPLOI. ILS ONT SERVI AUX AU
TEURS POURREALISER LES IMAGES DE CE JEU.
 VOUS TROUVEREZ, DANS 'AVENTURE SUR AMST
RAD', LES"
70 PRINT "EXPLICATIONS DETAILLEES DE CES
 PROGRAMMES, AINSI QUE LE MODE D'EMPLOI.
"
80 PRINT:PRINT:PRINT ">>>>>>>>>>>>>>>>
  ET MAINTENANT, FRAPPEZ UNE TOUCHE ET .
..     <<<<<<<<<<<<<<<<"
```

```
90 PRINT "..............................
   BON AMUSEMENT    ....................
........."
100 WHILE INKEY$="":WEND
1000 MEMORY 34380:DATA 33,0,0,205,5,188,
201,0,33,0,192,1,91,134,3,10,87,3,10,114
,35,61,194,104,166,125,238,255,194,99,16
6,124,238,255,194,99,166,201,0
1010 FOR I=34381 TO 34387:READ A:POKE I,
A:NEXT:FOR I=42588 TO 42619:READ A:POKE
I,A:NEXT
1020 MODE 1:LOAD "CIM0":INK 0,0:INK 1,0:
INK 2,0:INK 3,0:CALL 34381:CALL 42589:BO
RDER PEEK(65529):INK 0,PEEK(65530):INK 1
,PEEK(65531):INK 2,PEEK(65532):INK 3,PEE
K(65533):GOSUB 40000:RUN "AVENT.BAS"
40000 RESTORE 40040:E=0:ENV 1,3,-5,8:ENV
 2,1,-15,0
40010 FOR I=1 TO 2
40015 READ F1,D1,E1,F2,D2:IF F1=-1 THEN
40030 ELSE IF F1=-2 THEN I=2:E=1:GOTO 40
030
40020 IF INKEY$<>"" THEN RETURN ELSE SOU
ND 1,F1,D1,15,E1:SOUND 2,F2,D2,4:GOTO 40
015
40030 RESTORE 40040:NEXT:IF F1=-2 THEN 4
0010 ELSE IF E=1 THEN RETURN ELSE RESTOR
E 40060:GOTO 40010
40040 DATA 24,14,0,47,14,24,14,0,40,14,2
4,14,0,47,14,40,14,0,53,14,30,14,1,60,14
,0,14,2,80,14,20,14,1,60,14,0,14,2,47,14
,20,14,1,40,14,22,14,1,45,14,24,14,1,47,
14,27,14,1,53,14,24,14,1,47,14,0,14,2,40
,14,30,14,1,47,14,0,14,2,80,14,24,14,0,4
7,14,24,14,0
40050 DATA 80,14,24,14,0,63,14,40,14,0,5
3,14,30,14,1,60,14,0,14,2,80,14,20,14,1,
60,14,0,14,2,47,14,20,14,1,40,14,22,14,1
,80,14,24,14,1,63,14,27,14,1,53,14,30,56
,1,60,56,-1,1,1,1,1
```

```
40060 DATA 32,14,0,63,14,30,14,0,60,14,2
7,14,1,53,14,0,14,2,63,14,24,14,1,80,14,
0,14,2,53,14,27,14,1,80,14,0,14,2,63,14,
24,14,1,80,14,0,14,2,63,14,27,14,1,53,14
,30,14,1,47,14,32,14,1,53,14,30,14,1,47,
14,27,14,1,53,14,0,14,2,60,14,32,14,0,32
,14,30,14,0,30
40070 DATA 14,27,28,1,27,28,32,14,0,63,1
4,30,14,0,60,14,27,28,1,53,28,20,28,1,40
,28,21,14,1,45,14,20,14,1,53,14,18,14,1,
60,14,21,14,1,71,14,20,6,0,80,6,18,6,0,7
1,6,20,6,0,80,6,18,6,0,71,6,20,6,0,80,6,
18,6,0,71,6,20,6,0,80,6,18,6,0,71,6,20,8
,0,80,8
40080 DATA -2,1,1,1,1
```

PROGRAMME AVENT

N° Lignes		
0- 5	Initialisation	
10- 290	Analyse de syntaxe	
500- 880	Tests de direction	
1000- 1420	Description des lieux	
1500- 1590	Bateau	
2000- 2050	Dictionnaire	
2500- 2540	Calais	
3000- 3040	Douvres	
3500- 3660	Voir	(1)
4000- 4220	Prendre	(2)
4500- 4550	Sortir	(3)
5000- 5090	Ouvrir	(4)
5500- 5580	Fermer	(5)
6000- 6070	Attaquer	(6)
6500- 6710	Donner	(7)
7000- 7010	Laisser	(8)
7500- 7640	Descendre	(9) + Monter (10)
8000- 8170	Inventaire	(11)
8500- 8710	Entrer	(12)
9000- 9120	Demander	(13)
9500- 9550	Aller	(14)
10000-10200	Changer	(15)
10500-10540	Soigner	(16)
11000-11050	Ferrer	(17)
11500-11590	Payer	(18)
12000-12060	Appeler	(19)
12500-12550	Faire	(20)
13000-13080	Montrer	(21)
13500-13560	Frapper	(22)
14000-14060	Lire	(23)
14500-14540	Cueillir	(24)
15000-15020	Manger	(25)
15500-15520	Dormir	(26)
16000-16080	Acheter	(27)
16500-16510	Embrasser	(28)
17000-17700	Sauver	(33) + Reprendre (34)
45000-45010	Musique « petit navire »	
-50000	Appel prog. Fin	
64000-64400	Test + chargement images	

Liste des parties du programme principal.

```
5 POKE 65528,0:DIM VE$(34),NO$(35):EN=39
:C19%=1:C20%=1:C21%=1:C22%=1:C=17:C33%=1
:WINDOW #0,1,40,22,25:CLS:LOCATE 1,22:PR
INT "VOUS ETES MONSIEUR DE D'ARTAGNAN...
":GOTO 2000
10 C0%=C0%+1:K=0:INPUT"-->",R$:R$=UPPER$
(R$)
20 IF LEN(R$)=0 THEN 10
30 IF R$="I" THEN 8000
40 IF LEN(R$)=1 THEN 500
50 IF RIGHT$(R$,1)=" " THEN 70
60 R$=R$+" "
70 VB=0:VR$=""
80 RE$="":NM=0:R=0
90 A=R+1:R=INSTR(A,R$," ")
100 RE$=MID$(R$,A,R-A)
110 IF MID$(RE$,2,1)="'" THEN RE$=MID$(R
E$,3)
115 IF K=0 THEN V$=RE$ ELSE N$=RE$
120 IF LEN(RE$)>6 THEN RE$=LEFT$(RE$,6):
 GOTO 150
130 RP=ASC(RIGHT$(RE$,1))
140 IF RP<65 OR RP>90 THEN RE$=LEFT$(RE$
,LEN(RE$)-1)
150 IF K=1 THEN 220
160 FOR I=1 TO NV
170 IF VE$(I)=RE$ THEN VB=I:VR$=VE$(I):I
=NV:K=1
180 NEXT I:IF R=LEN(R$) AND VB=0 THEN IF
 C33%=1 THEN PRINT "JE NE COMPREND PAS":
 GOTO 10 ELSE PRINT "I DON'T UNDERSTAND"
:GOTO 10
190 IF VB=11 THEN 8000
195 IF VB=33 OR VB=34 THEN 17000
200 IF VB>28 THEN R$=LEFT$(VR$,1):GOTO 5
00
210 IF K=1 THEN 80 ELSE 90
220 FOR I=1 TO NN
230 IF NO$(I)=VR$ THEN 250
240 IF NO$(I)=RE$ THEN NM=I:I=NN
250 NEXT I
260 IF R<>LEN(R$) AND NM=0 THEN 90
```

```
270 IF NM=1 THEN 8000
280 IF NM>1 AND NM<6 THEN R$=LEFT$(RE$,1
):GOTO 500
290 ON VB GOTO 3500,4000,4500,5000,5500,
6000,6500,7000,7500,7500,8000,8500,9000,
9500,10000,10500,11000,11500,12000,12500
,13000,13500,14000,14500,15000,15500,160
00,16500
500 IF R$<>"N" AND R$<>"S" AND R$<>"E" A
ND R$<>"O" AND R$<>"W" OR R$="O" AND C33
%=0 OR R$="W" AND C33%=1 THEN 860
510 IF C>16 AND C<31 THEN IF C33%=1 THEN
 PRINT "VOUS ETES A L'INTERIEUR D'UNE PI
ECE":GOTO 10 ELSE PRINT "YOU ARE INSIDE
A ROOM":GOTO 10
520 IF C>9 AND C<17 AND C<>14 OR C>33 TH
EN GOSUB 810:IF K=0 THEN 10
530 ON C GOTO 540,550,570,580,590,590,60
0,570,570,610,620,630,640,580,650,660,51
0,510,510,510,510,510,510,510,510,510,51
0,510,510,510,670,700,720,730,740,750,76
0,780
540 IF R$="N" THEN C=5:GOTO 1000 ELSE IF
 R$="S" THEN C=6:GOTO 1000 ELSE IF R$="E
" THEN C=7:GOTO 1000 ELSE 790
550 IF R$="E" THEN GOSUB 810:IF K=1 THEN
 C=33:GOTO 1000 ELSE 10
560 IF R$="O" THEN C=7:GOTO 1000 ELSE 79
0
570 IF C=3 AND R$="O" OR C=9 AND R$="S"
OR C=8 AND R$="E" THEN C=31:GOTO 1000 EL
SE 790
580 IF C=4 AND R$="W" OR C=14 AND R$="S"
 THEN C=32:GOTO 1000 ELSE 800
590 IF C=5 AND R$="S" OR C=6 AND R$="N"
THEN C=1:GOTO 1000 ELSE 790
600 IF R$="O" THEN C=1:GOTO 1000 ELSE IF
 R$="E" THEN C=2:GOTO 1000 ELSE 790
610 IF R$="E" THEN C=11:GOTO 1000 ELSE 8
00
620 IF R$="N" THEN C=35:GOTO 1000 ELSE I
F R$="W" THEN C=10:GOTO 1000 ELSE 800
```

```
630 IF R$="S" THEN C=36:GOTO 1000 ELSE 8
00
640 IF R$="N" THEN C=37:GOTO 1000 ELSE 8
00
650 IF R$="S" THEN C=33:GOTO 1000 ELSE 7
90
660 IF R$="W" THEN C=37:GOTO 1000 ELSE 8
00
670 IF C26%=1 THEN PRINT "D'ACCORD, MAIS
 PAS A CHEVAL":GOTO 10
680 IF R$="E" THEN IF C37%<>0 THEN C=3:G
OTO 1000 ELSE PRINT "UN GARDE VOUS EMPEC
HE DE PASSER":GOTO 10
690 IF R$="N" THEN C=9:GOTO 1000 ELSE IF
 R$="O" THEN C=8:GOTO 1000 ELSE 790
700 IF C26%=1 THEN PRINT "OK, BUT NOT WI
TH YOUR HORSE":GOTO 10
710 IF R$="E" THEN C=4:GOTO 1000 ELSE IF
 R$="N" THEN C=14:GOTO 1000 ELSE 800
720 IF R$="N" THEN C=15:GOTO 1000 ELSE I
F R$="E" THEN 2500 ELSE IF R$="S" THEN C
=38:GOTO 1000 ELSE IF R$="O" THEN C=2:GO
TO 1000 ELSE 860
730 IF R$="O" THEN C=33:GOTO 1000 ELSE 7
90
740 IF R$="E" THEN C=36:GOTO 1000 ELSE I
F R$="S" THEN C=11:GOTO 1000 ELSE 800
750 IF R$="N" THEN C=12:GOTO 1000 ELSE I
F R$="E" THEN C=37:GOTO 1000 ELSE IF R$=
"W" GOTO 3000 ELSE 800
760 IF R$="E" THEN IF C29%=1 AND C24%=1
THEN C=16:GOTO 1000 ELSE IF C24%=0 THEN
PRINT "YOU'RE LOST IN LONDON... TOO LATE
":GOTO 50000  ELSE IF C29%=0 THEN PRINT
"A BOBBY ASK YOUR RESIDENCE PERMIT":GOTO
 10
770 IF R$="S" THEN C=13:GOTO 1000 ELSE I
F R$="W" THEN C=36:GOTO 1000 ELSE 800
780 IF R$="N" THEN C=33:GOTO 1000
790 PRINT "PAS PAR LA : IL FAUT BIEN DES
 LIMITES   AU JEU !":GOTO 10
```

```
800 PRINT "NOT IN THAT DIRECTION : THIS
GAME REQUI-RES LIMITS !":GOTO 10
810 K=0:IF C26%=0 THEN IF C33%=1 THEN PR
INT "A PIED VOUS PERDEZ DU TEMPS":RETURN
 ELSE PRINT "WALKING ? YOU'LL WASTE TIME
":RETURN
820 K=1:RETURN
860 IF C33%=1 THEN PRINT R$;" N'EST PAS
UNE DIRECTION":GOTO 880
870 PRINT R$;" IS NOT A DIRECTION"
880 R$="":GOTO 10
1000 GOSUB 64000
1005 ON C GOTO 1010,1020,1030,1040,1050,
1060,1070,1080,1090,1100,1110,1120,1130,
1140,1150,1160,1170,1180,1190,1200,1210,
1260,1270,1280,1290,1300,1310,1320,1420,
1330,1340,1350,1360,1370,1380,1390,1400,
1410
1010 PRINT "RUE DE TROUSSE-CHEMISE,":PRI
NT "DEVANT VOTRE CHAMBRE":GOTO 10
1020 PRINT "ICI HABITE UNE CHARMANTE VEU
VE...":GOTO 10
1030 PRINT "LA PORTE DU BOUDOIR DE LA RE
INE.":GOTO 10
1040 PRINT "HERE IS THE DOOR OF THE DUKE
'S LIBRARY":GOTO 10
1050 PRINT "VOICI L'ENTREE DE VOTRE ECUR
IE...":GOTO 10
1060 PRINT "L'ECHOPE DE L'USURIER...":PR
INT "DONNEZ BEAUCOUP POUR AVOIR PEU":GOT
O 10
1070 PRINT "L'ENTREE DE L'ESTAMINET DU C
OIN,":PRINT "RENDEZ-VOUS PRIVILEGIE DES
GARDES ET DESMOUSQUETAIRES":GOTO 10
1080 PRINT "LE FUMET QUI S'ECHAPPE DE CE
TTE CUISINE VOUS EXCITE LES NARINES":GOT
O 10
1090 PRINT "PALSAMBLEU ! VOUS ETES DEVAN
T LA SALLE  DES GARDES DU CARDINAL":GOTO
 10
1100 PRINT "YOU CAN SEE THE ENTRANCE OF
A RESTAURANT":GOTO 10
```

```
1110 PRINT "YOU ARE IN FRONT OF THE CUST
OM HOUSE":GOTO 10
1120 PRINT "A LITTLE BARN AND, INSIDE, W
ELCOMING HAY":GOTO 10
1130 PRINT "YOU ARE STANDING IN FRONT OF
 A BOOKSHOP":GOTO 10
1140 PRINT "PINED ON THIS DOOR, A PANNEL
 INDICATE  'SECRETARIAT OF THE DUKE'":G
OTO 10
1150 PRINT "VOICI L'ENTREE DU PALAIS DE
VERSAILLES":GOTO 10
1160 PRINT "HERE WE ARE ! THE ENTRANCE O
F BUCKINGHAM PALACE":GOTO 10
1170 PRINT "DANS VOTRE CHAMBRE IL Y A :
UN COFFRE,  UNE CHEMINEE, UNE PORTE, UNE
 ARMOIRE,  UN BUREAU, UNE PATERE":GOTO
10
1180 PRINT "WAOUH ! SI JEUNE, SI BELLE..
.":PRINT "ET POURTANT SI VEUVE...":GOTO
10
1190 PRINT "VOUS ETES DEVANT LA REINE...
":GOTO 10
1200 PRINT "HALF OF YOUR TRIP IS ACHIEVE
D :":PRINT "HERE IS THE DUKE OF BUCKINGH
AM":GOTO 10
1210 CLS:PRINT "           VOICI VOS CHE
VAUX :":IF C19%=1 THEN LOCATE 5,2:PRINT
"BONAMI"
1220 IF C20%=1 THEN LOCATE 13,2:PRINT "F
LAMBEAU"
1230 IF C21%=1 THEN LOCATE 22,2:PRINT "L
AFLECHE"
1240 IF C22%=1 THEN LOCATE 31,2:PRINT "M
IRANDOLE"
1250 GOTO 10
1260 PRINT "L'USURIER S'APPELLE MATANTE"
:GOTO 10
1270 PRINT "LA FUMEE LAISSE DEVINER LE P
ATRON ET DESGENS ATTABLES A GAUCHE ";:IF
 C16%=0 THEN PRINT "ET A DROITE":GOTO 10
 ELSE GOTO 10
```

```
1280 PRINT "QUE DE BONNES CHOSES DANS CE
S MARMITES":GOTO 10
1290 PRINT "LES GARDES DU CARDINAL SONT
TRES TRES   MENACANTS":GOTO 10
1300 PRINT "INSIDE THE RESTAURANT, THE O
WNER":PRINT "WELCOMES YOU":GOTO 10
1310 PRINT "OK, THE USUAL FORMALITIES AR
E DONE. IF  YOU NEED ANYTHING ELSE PLEAS
E COME BACK":GOTO 10
1320 PRINT "THE HAY IS SOFT AND FRESH...
":GOTO 10
1330 PRINT "THIS SMASHING BABBYSOXER IS
SUPPOSED TO BE A SECRETARY":GOTO 10
1340 PRINT "VOUS N'AVEZ PAS LE TEMPS D'A
DMIRER LA   COUR DU CHATEAU":GOTO 10
1350 PRINT "THE YARD OF THE PALACE. FRIS
KY HORSES   ARE WAITING...":GOTO 10
1360 PRINT "DOUCE CAMPAGNE PARSEMEE DE F
LEURS.      IL Y A UN PANNEAU":GOTO 10
1370 PRINT "VOICI LE PORT DE CALAIS.":GO
TO 10
1380 PRINT "YOU ARE ON THE DOVER'S PIER.
":GOTO 10
1390 PRINT "WHAT A QUIET FIELD ! PRETTY
FLOWERS     WOULD BE A TENDER GIFT":GOTO
 10
1400 PRINT "A STREET OF LONDON. A COSTER
MONGER SELLSFLOWERS.":GOTO 10
1410 PRINT "LE RELAIS DE LA JUMENT BLEUE
. LA STATIONSERVICE DE L'EPOQUE. LE PATR
ON TRAVAIL- LE. UN PANNEAU EST ACCROCHE
A LA PORTE":GOTO 10
1420 PRINT "THE BOOKSELLER PROPOSE : PEN
CILS, BOOKS,MAPS, NEWSPAPERS....":GOTO 1
0
1500 IF C=35 THEN 1560
1510 IF C35%=0 THEN PRINT "LE CAPITAINE
N'EST PAS LA":GOTO 10
1520 IF C13%=0 THEN 16080 ELSE IF C13%<3
00 THEN 16060
1540 C13%=C13%-300:PRINT "TOUT LE MONDE
EMBARQUE, ON LEVE L'ANCRE ET...":IM$="CI
```

```
MMER":GOSUB 64100:GOSUB 45000
1545 IF C16%=0 THEN PRIN
T "VOUS ETES SEUL ET L'EQUIPAGE VOUS VOL
E.":C32%=0:C13%=0:C15%=0
1550 IF C9%<>1 THEN PRINT "... VOUS MOUR
REZ D'INSOLATION !":GOTO 50000 ELSE C=35
:C33%=0:C35%=0:C26%=0:GOTO 2000
1560 IF C35%=0 THEN PRINT "THE CAPTAIN I
S NOT HERE":GOTO 10
1570 IF C15%=0 THEN 16070
1580 IF C15%<30 THEN 16050
1590 C15%=C15%-30:PRINT "EVERYBODY IS ON
 BOARD AND...":IM$="CIMMER":GOSUB 64100:
GOSUB 45000:C=34:C33%=1:C35%=0:C26%=0
2000 NV=34:IF C33%=1 THEN NN=35:RESTORE
2020 ELSE NN=18:RESTORE 2040
2010 FOR I=1 TO NV:READ VE$(I):NEXT I:FO
R I=1 TO NN:READ NO$(I):NEXT I:GOTO 1000
2020 DATA VOIR,PRENDR,SORTIR,OUVRIR,FERM
ER,ATTAQU,DONNER,LAISSE,DESCEN,MONTER,IN
VENT,ENTRER,DEMAND,ALLER,CHANGE,SOIGNE,F
ERRER,PAYER,APPELE,FAIRE,MONTRE,FRAPPE,L
IRE,CUEILL,MANGER,DORMIR,ACHETE,EMBRAS,N
ORD,SUD,EST,OUEST,SAUVER,REPREN
2030 DATA INVENT,NORD,SUD,EST,OUEST,CONV
OC,CHEVAL,FLEURS,ARGENT,LIVRES,REVERE,CA
PITA,BATEAU,PORTE,CHAMBR,CHEMIN,BUREAU,A
RMOIR,COFFRE,TIROIR,PATERE,CAPE,CHAPEA,S
AUF-C,EPEE,BIJOUX,PANNEA,PISTOL,DROITE,G
AUCHE,PATRON,BONAMI,FLAMBE,LAFLEC,MIRAND
2040 DATA LOOK,GET,OUT,OPEN,SHUT,ATTACK,
GIVE,LET,DOWN,UP,INVENT,IN,ASK,GO,CHANGE
,TREAT,SHOE,PAY,CALL,DO,SHOW,KNOCK,READ,
PICK,EAT,SLEEP,BUY,KISS,NORTH,SOUTH,EAST
,WEST,SAVE,RESTOR
2050 DATA INVENT,NORTH,SOUTH,EAST,WEST,R
ESIDE,HORSE,FLOWER,MONEY,POUNDS,BOW,CAPT
AI,BOAT,DOOR,DUKE,RING,MAP,FERRET
2500 IF C11%=1 THEN PRINT "BONAMI ETAIT
MALADE ET IL MEURT...":GOTO 50000
2510 IF C11%=2 THEN PRINT "FLAMBEAU ETAI
T TROP LENT...":GOTO 50000
```

```
2520 IF C11%=4 THEN PRINT "MIRANDOLE ETA
IT MAL FERRE. IL SE CASSE  UNE JAMBE..."
:GOTO 50000
2530 IF C0%>90 THEN PRINT "VOUS AVEZ PER
DU TROP DE TEMPS...":GOTO 50000
2540 C=34:GOTO 1000
3000 IF C11%<6 THEN PRINT "SINCE FRANCE,
 YOU RIDE THE SAME HORSE.  HE DEAD TIRED
 OUT... TOO LATE":GOTO 50000
3010 IF C27%=0 THEN PRINT "SINCE THE BEG
INNING WITHOUT SLEEPING !  YOU ARE COMPL
ETELY EXHAUSTED AND YOU    SLEEP DURING
THREE DAYS... TOO LATE":GOTO 50000
3020 IF C28%=0 THEN PRINT "SINCE YOUR DE
PARTURE YOU HAVE NOT EAT.  YOU FALL SICK
 WITH STOMACH CRAMPS...    TOO LATE !":G
OTO 50000
3030 IF C0%>200 THEN PRINT "YOU'VE WASTE
D TOO MUCH TIME...":GOTO 50000
3040 C=35:GOTO 1000
3500 IF NM=0 THEN 1005
3505 IF C=17 THEN 3510 ELSE IF C=23 THEN
 3640 ELSE IF C=33 OR C=38 GOTO 3650 ELS
E IF C=30 THEN 3660 ELSE GOTO 4010
3510 IF NM>21 OR NM<14 THEN PRINT "A PRE
MIERE VUE JE NE VOIS PAS CELA ICI":GOTO
10
3520 IF NM=15 THEN GOTO 1000 ELSE IF NM=
16 THEN PRINT "ELLE A ETE RAMONNEE IL Y
A PEU DE TEMPS":GOTO 10 ELSE IF NM=17 TH
EN 3580
3530 IF NM =18 THEN IF C6%=0 THEN 3650 E
LSE IF C7%>0 THEN 3630 ELSE PRINT "IL Y
A UNE CAPE":GOTO 10
3540 IF NM =19 THEN IF C8%=0 THEN 3650 E
LSE IF C9%>0 THEN 3630 ELSE PRINT "IL Y
A UN CHAPEAU":GOTO 10
3550 IF NM =20 THEN IF C4%=0 THEN 3650 E
LSE IF C5%>0 THEN 3630 ELSE PRINT "IL Y
A DES BIJOUX":GOTO 10
3560 IF NM =21 THEN IF C10%>0 THEN 3630
ELSE PRINT "VOTRE EPEE Y EST PENDUE":GOT
O 10
```

```
3570 IF NM =14 THEN IF C34%=0 THEN 3650
ELSE PRINT "ELLE EST OUVERTE ET VOS CREA
NCIERS SONT TOUJOURS LA":GOTO 10
3580 C1%=1:PRINT "SUR LE DESSUS ON PEUT
VOIR :"
3590 IF C2%=0 THEN PRINT "UNE CONVOCATIO
N,";:IF C3%=1 THEN 10
3600 IF C3%=0 THEN PRINT "UN SAUF-CONDUI
T":GOTO 10
3610 PRINT "RIEN PUISQUE VOUS AVEZ TOUT
PRIS":GOTO 10
3630 PRINT "C'EST PLEIN DE VIDE !":GOTO
10
3640 IF NM=31 THEN 12000 ELSE PRINT "LA
FUMEE VOUS EMPECHE DE VOIR.":PRINT "DE Q
UEL COTE VOULEZ VOUS ALLER ?":GOTO 10
3650 IF NM=27 THEN 14000 ELSE PRINT "BEN
... IL N'Y A RIEN A EN DIRE":GOTO 10
3660 IF NM=15 THEN IF C38%=1 THEN PRINT
"I TOLD YOU THE DUKE IS IN HIS LIBRARY":
GOTO 10 ELSE IF NM=15 THEN PRINT "SORRY,
 HE'S IN CONFERENCE":GOTO 10 ELSE PRINT
"NOTHING IS NOTEWORTHY":GOTO 10
4000 IF NM=0 THEN 4210
4005 IF C=17 THEN 4030 ELSE IF C=21 THEN
 4140 ELSE IF C=33 OR C=36 THEN 4200
4010 IF C33%=1 THEN PRINT "NON ! ICI, IL
 N'Y A RIEN A ";V$:GOTO 10
4020 PRINT "NO ! HERE, THERE'S NOTHING T
O ";V$:GOTO 10
4030 IF NM=6 THEN IF C1%=0 THEN 4130 ELS
E IF C2%=1 THEN 4110 ELSE C2%=1: GOTO 41
20
4040 IF NM=24 THEN IF C1%=0 THEN 4130 EL
SE IF C3%=1 THEN 4110 ELSE C3%=1:GOTO 41
20
4050 IF NM=25 THEN IF C10%<>0 THEN 4110
ELSE C10%=1:GOTO 4120
4060 IF NM=26 THEN IF C4%=0 THEN 4130 EL
SE IF C5%<>0 THEN 4110 ELSE C5%=1:C4%=0:
GOTO 4120
```

```
4070 IF NM=23 THEN IF C8%=0 THEN 4130 EL
SE IF C9%<>0 THEN 4110 ELSE C9%=1:C8%=0:
GOTO 4120
4080 IF NM=22 THEN IF C6%=0 THEN 4130 EL
SE IF C7%<>0 THEN 4110 ELSE C7%=1:C6%=0:
GOTO 4120
4100 PRINT "VOUS NE POUVEZ PAS ";V$;" CE
LA":GOTO 10
4105 PRINT "YOU CAN'T ";V$;" THAT":GOTO
10
4110 PRINT "VOUS AVEZ DEJA CELA":GOTO 10
4115 PRINT "YOU'VE ALREADY GOT ONE":GOTO
 10
4120 PRINT "VOUS AVEZ RAISON, CELA PEUT
SERVIR":GOTO 1000
4130 PRINT "D'ACCORD, MAIS OU ?":GOTO 10
4140 IF C11%<>0 THEN 4110
4150 IF NM=32 THEN C19%=0:C11%=1:GOTO 41
85
4160 IF NM=33 THEN C20%=0:C11%=2:GOTO 41
85
4170 IF NM=34 THEN C21%=0:C11%=3:GOTO 41
85
4180 IF NM=35 THEN C22%=0:C11%=4
4185 IF C11%<>0 THEN GOSUB 64000:PRINT "
VOUS AVEZ ";N$:GOTO 10
4190 PRINT "QUEL CHEVAL VOULEZ-VOUS ?":G
OTO 1000
4200 IF NM=8 THEN 14500 ELSE GOTO 4010
4210 IF C33%=1 THEN PRINT "PRECISEZ CE Q
UE VOUS VOULEZ ";V$:GOTO 10
4220 PRINT "TELL ME WHAT YOU WANT TO ";V
$:GOTO 10
4500 IF C=17 THEN 4530 ELSE IF C=19 OR C
=20 THEN C30%=0:C31%=0:C37%=0:C38%=0:GOT
O 4520 ELSE IF C=23 THEN 4550
4510 IF C<17 OR C>33 THEN 8560
4520 C=C-16:GOTO 1000
4530 IF NM=16 THEN C1%=0:GOTO 4520 ELSE
IF C34%=0 THEN PRINT "VOUS N'ETES PAS LE
 PASSE-MURAILLE":GOTO 10
4540 PRINT "UNE ARMEE DE CREANCIERS VOUS
```

```
ATTEND      DEVANT VOTRE PORTE":GOTO 10
4550 IF C16%=0 OR C18%=1 THEN C17%=0:GOT
O 4520 ELSE C17%=1:PRINT "NON ! DIT LE P
ATRON, VOUS DEVEZ PAYER   LES CONSOMMATI
ONS":GOTO 10
5000 IF C<>17 THEN 4010
5010 IF NM=0 THEN 4210
5020 IF NM=14 THEN IF C34%=1 THEN 5070 E
LSE C34%=1:GOTO 5090
5025 IF NM<18 OR NM>20 THEN 4100 ELSE JL
=NM-17:ON JL GOTO 5030,5040,5050
5030 IF C6%=1 THEN 5070 ELSE IF C7%=1 TH
EN 5060 ELSE C6%=1:C4%=0:C8%=0:GOTO 5080
5040 IF C8%=1 THEN 5070 ELSE IF C9%=1 TH
EN 5060 ELSE C8%=1:C4%=0:C6%=0:GOTO 5080
5050 IF C4%=1 THEN 5070 ELSE IF C5%=1 TH
EN 5060 ELSE C4%=1:C6%=0:C8%=0:GOTO 5080
5060 PRINT "CE N'EST PAS LA PEINE : ";:G
OTO 3630
5070 PRINT "MAIS C'EST DEJA OUVERT":GOTO
 10
5080 GOSUB 64000
5090 PRINT "VOILA C'EST OUVERT":GOTO 10
5500 IF C<>17 THEN 4010
5510 IF NM=0 THEN 4210
5520 IF NM<18 AND NM<>14 OR NM>20 THEN 4
100
5530 IF NM=19 AND C8%=1 THEN C8%=0:GOTO
5580
5540 IF NM=14 AND C34%=1 THEN C34%=0:GOT
O 5580
5550 IF NM=18 AND C6%=1 THEN C6%=0:GOTO
5580
5560 IF NM=20 AND C4%=1 THEN C4%=0:GOTO
5580
5570 PRINT "MAIS C'EST DEJA FERME":GOTO
10
5580 GOSUB 64000:PRINT "VOILA C'EST FERM
E":GOTO 10
6000 IF C=17 THEN 6020 ELSE IF C=23 OR C
=25 THEN 6050
```

```
6010 IF C33%=1 THEN PRINT "ALLONS, ALLON
S... RESTONS CALME":GOTO 10 ELSE IF C33%
=0 THEN PRINT "COOL, BUDDY, KEEP COOL":G
OTO 10
6020 IF C34%=0 THEN 6010
6030 IF C10%<>1 THEN IM$="CIMPRIS":GOSUB
 64100:PRINT "VOUS N'AVEZ PAS D'ARME ET
ILS VOUS FONT ARRETER...":GOTO 50000
6040 GOSUB 64000:PRINT "GRACE A VOTRE EP
EE,":PRINT "ILS NE PEUVENT QUE VOUS REPO
USSER":GOTO 1000
6050 IF C10%=1 AND C16%=1 THEN GOSUB 640
00:PRINT "ILS VOUS REPOUSSENT":GOTO 1000
6060 IF C10%=1 OR C16%=1 THEN C0%=C0%+15
:IM$="CIMSOIN":GOSUB 64100:PRINT "ILS VO
US BLESSENT":PRINT "VOUS PERDEZ DU TEMPS
 A VOUS SOIGNER":GOTO 1000
6070 IM$="CIMPRIS":GOSUB 64100:PRINT "SE
UL ET SANS ARMES,":PRINT "VOUS VOUS FAIT
ES ARRETER...":GOTO 50000
6500 IF NM=0 THEN 4210
6520 IF C=17 OR C=1 THEN 6560 ELSE IF C=
22 THEN 10090 ELSE IF C=30 THEN 6670
6530 IF C=18 THEN PRINT "NE LUI DONNEZ R
IEN":PRINT "VOUS RISQUERIEZ DE LA VEXER"
:GOTO 10
6540 IF NM=9 OR NM=10 OR NM=28 THEN 1150
0
6550 IF C33%=1 THEN PRINT "NON ! VOUS RI
QUERIEZ D'EN AVOIR BESOIN":GOTO 10:ELSE
PRINT "NO ! PERHAPS IT'LL BE USEFULL LAT
ER":GOTO 10
6560 IF NM=22 THEN IF C7%<>1 THEN 6640 E
LSE C7%=2:GOTO 6650
6570 IF NM=23 THEN IF C9%<>1 THEN 6640 E
LSE C9%=2:GOTO 6650
6580 IF NM=25 THEN IF C10%<>1 THEN 6640
ELSE C10%=2:GOTO 6650
6590 IF NM=26 THEN IF C5%<>1 THEN 6640 E
LSE C5%=2:GOTO 6650
6600 IF NM=9 THEN IF C13%=0 AND C15%=0 T
HEN 6640 ELSE C13%=0:C15%=0:GOTO 6650
```

```
6610 IF NM=28 THEN IF C13%=0 THEN 6640 E
LSE C13%=0:GOTO 6650
6620 IF NM=10 THEN IF C15%=0 THEN 6640 E
LSE C15%=0:GOTO 6650
6630 PRINT "CELA NE LES INTERRESSE PAS":
GOTO 10
6640 PRINT N$+" ? POUR L'INSTANT VOUS N'
EN AVEZ PAS !":GOTO 10
6645 PRINT N$+" ? IN THAT TIME, YOU HAVE
 NONE":GOTO 10
6650 PRINT "CELA NE SUFFIT PAS...":PRINT
 "POUR EPONGER VOS DETTES":GOTO 10
6670 IF NM<>8 THEN 6550
6680 IF C38%=1 THEN PRINT "YOU HAVE ALRE
ADY GIVEN FLOWERS":GOTO 10
6690 IF C36%=1 THEN C38%=1:IM$=IM$+"P":G
OSUB 64100:PRINT "THEY'VE TURNED TO FADE
.":PRINT "SECRETARY IS HURTED":GOTO 10
6700 IF C36%>1 THEN C38%=1:IM$="CIM30F":
GOSUB 64100:PRINT "GRATEFULL, SHE GIVES
YOU A TENDER PECK. THE DUKE IS WAITING F
OR YOU.":C36%=0:GOTO 10
6710 GOTO 6645
7000 IF C<>21 THEN IF C33%=1 THEN 4100 E
LSE 4105
7010 IF NM=0 THEN 4210 ELSE GOTO 10030
7500 IF C=3 OR C=8 OR C=9 THEN 6640
7510 IF C=4 OR C=14 THEN 6645
7520 IF C11%=0 THEN IF C33%=1 THEN PRINT
 "VOUS N'AVEZ PAS DE CHEVAL":GOTO 10 ELS
E PRINT "YOU HAVE NO HORSE":GOTO 10
7530 IF C>16 AND C<31 AND C33%=1 THEN PR
INT "NON, PAS A L'INTERIEUR D'UNE PIECE.
":GOTO 10
7540 IF C>16 AND C<31 AND C33%=0 THEN PR
INT "NOT INSIDE A ROOM : THERE IS NEITHE
R     STAIRCASE OR HORSE":GOTO 10
7550 IF (C=34 OR C=35) AND NM=13 THEN 76
20
7560 IF C33%=1 AND (C26%=0 AND VB=9 OR C
26%=1 AND VB=10) THEN PRINT "JE VOUS SIG
```

```
NALE QUE VOUS ETES DEJA":IF VB=9 THEN PR
INT "A PIED":GOTO 10 ELSE PRINT "A CHEVA
L":GOTO 10
7570 IF C33%=0 AND (C26%=0 AND VB=9 OR C
26%=1 AND VB=10) THEN PRINT "BUT YOU ARE
 ALREADY ";:IF VB=9 THEN PRINT "WALKING"
:GOTO 10 ELSE PRINT "RIDING":GOTO 10
7580 IF VB=9 AND C33%=1 THEN C26%=0:PRIN
T "EH BIEN, MARCHEZ MAINTENANT":GOTO 10
7590 IF VB=9 AND C33%=0 THEN C26%=0:PRIN
T "WELL, YOU CAN WALK NOW":GOTO 10
7600 IF VB=10 AND C33%=1 THEN C26%=1:PRI
NT "EH BIEN TROTTEZ MAINTENANT":GOTO 10
7610 IF VB=10 AND C33%=0 THEN C26%=1:PRI
NT "WELL, YOU CAN TROT NOW":GOTO 10
7620 IF C=35 THEN 7640
7630 IF C35%=0 THEN PRINT "LE CAPITAINE
N'EST PAS LA POUR VOUS EN  DONNER L'AUTO
RISATION":GOTO 10 ELSE PRINT "IL FAUT PA
YER LE PRIX":GOTO 10
7640 IF C35%=0 THEN PRINT "THE CAPTAIN I
S NOT HERE TO GIVE THE     PERMISSION":G
OTO 10 ELSE PRINT "YOU MUST PAY THE PRIC
E":GOTO 10
8000 WINDOW #0,1,40,1,25:CLS:PRINT "VOUS
 POSSEDEZ :":I=0
8010 IF C2%=1 THEN PRINT "LA CONVOCATION
 DE LA REINE":I=1
8020 IF C3%=1 THEN PRINT "UN SAUF-CONDUI
T":I=1
8030 IF C5%=1 THEN PRINT "LES BIJOUX":I=
1
8040 IF C7%=1 THEN PRINT "VOTRE CAPE":I=
1
8050 IF C9%=1 THEN PRINT "VOTRE CHAPEAU"
:I=1
8060 IF C10%=1 THEN PRINT "VOTRE EPEE":I
=1
8070 IF C13%<>0 THEN PRINT C13%;" PISTOL
ES":I=1
```

```
8080 IF C15%<>0 THEN PRINT C15%;" LIVRE
S":I=1
8090 IF C16%=1 THEN PRINT "ATHOS, PORTOS
, ARAMIS":I=1
8100 IF C32%=1 THEN PRINT "LA BAGUE DE L
A REINE":I=1
8110 IF C36%<>0 THEN PRINT "UN BOUQUET D
E FLEURS":I=1
8120 IF C29%=1 THEN PRINT "A RESIDENCE P
ERMIT":I=1
8130 IF C24%=1 THEN PRINT "A MAP OF LOND
ON":I=1
8140 IF C12%=1 THEN PRINT "LES FERRETS D
E LA REINE":I=1
8150 IF C11%<>0 THEN PRINT "UN CHEVAL":I
=1
8160 IF I=0 THEN PRINT "J'AI BEAU FOUILL
ER VOTRE BESACE...":PRINT "JE NE VOIS RI
EN"
8170 IF INKEY$="" THEN 8170 ELSE CALL 34
381:CALL 42589:WINDOW #0,1,40,22,25:CLS:
GOTO 1005
8500 IF C=15 OR C=16 THEN 8550
8510 K=-(C<5)-2*(C>4 AND C<15)-3*(C>16 A
ND C<31)-4*(C>30 AND C<38)-5*(C=38)
8520 ON K GOTO 8610,8530,8560,8580,8600
8530 IF C26%=1 AND C33%=1 THEN PRINT "ET
 SI ON ENTRAIT CHEZ VOUS A CHEVAL,":PRIN
T "QU'EN PENSERAIT VOTRE TAPIS PERSAN ?"
:GOTO 10
8540 IF C26%=1 AND C33%=0 THEN PRINT "AN
D WHAT ABOUT YOUR PERSAN CARPET...":PRIN
T "IF SOMEBODY WAS RIDING IN YOUR HOME ?
":GOTO 10
8550 C=C+16:GOTO 1000
8560 IF C33%=1 THEN PRINT "MAIS VOUS ETE
S DEJA A L'INTERIEUR":GOTO 10
8570 PRINT "BUT YOU ARE ALREADY INSIDE":
GOTO 10
8580 IF C33%=1 THEN PRINT "DANS QUELLE D
IRECTION ?":GOTO 10
8590 PRINT "WHICH DIRECTION ?":GOTO 10
```

```
8600 PRINT "C'EST PRIVE, IL EST INTERDIT
 D'Y ENTRER":GOTO 10
8610 ON C GOTO 8620,8650,8670,8700
8620 IF C26%=1 THEN PRINT "PAS A CHEVAL,
":PRINT "UN PEU DE TENUE TOUT DE MEME !"
:GOTO 10
8630 IF NM<>16 THEN PRINT "ATTENTION !":
PRINT "LES CREANCIERS SONT TOUJOURS LA..
.":GOTO 10
8640 GOTO 8550
8650 IF C26%=1 THEN PRINT "DITES,":PRINT
 "ON NE RENTRE PAS A CHEVAL CHEZ LES GEN
S":GOTO 10
8660 C0%=C0%+20:GOTO 8550
8670 IF C30%=0 THEN PRINT "ALLONS, ALLON
S...":PRINT "SI LA REINE ETAIT EN GALANT
E COMPAGNIE ?":GOTO 10
8680 IF C12%=1 THEN C=C+16:IM$="CIM19":G
OSUB 64000:POKE 65528,1:PRINT "MONSIEUR
 D'ARTAGAN !":PRINT "VOUS AVEZ REUSSI VOT
RE MISSION !":PRINT "JE VOUS SERAIS RECO
NNAISSANTE A JAMAIS !":GOTO 50000
8690 IF C32%=1 THEN PRINT "JE N'AI PLUS
RIEN A VOUS DIRE.":PRINT "VOUS PERDEZ VO
TRE TEMPS":GOTO 10 ELSE GOTO 8550
8700 IF C30%=0 THEN PRINT "OH ! SHOCKING
 !":PRINT "PERHAPS THE DUKE IS BARE !":G
OTO 10
8710 IF C12%=1 THEN PRINT "SORRY, I CAN'
T TELL YOU ANYTHING MORE.  YOU HAVE NO T
IME TO SPARE":C=C+16:GOTO 10 ELSE 8550
9000 IF C<>20 AND C<>27 AND C<>29 THEN 4
010
9010 IF C=29 THEN 9060 ELSE IF C=27 THEN
 9100
9020 IF C31%=0 THEN PRINT "YOU ARE VERY
IMPOLITE, MUSKETEER !":GOTO 10
9030 IF NM<>18 THEN PRINT "SORRY, I CAN'
T GIVE YOU THAT":GOTO 10
9040 IF C23%=0 THEN PRINT "SORRY, MONSIE
UR, I DON'T KNOW YOU":GOTO 10
```

```
9050 PRINT "OK, HERE ARE THE FERRETS.":P
RINT "PLEASE, GIVE THE QUEEN MY BEST REG
ARDS":C12%=1:GOTO 10
9060 IF NM<>17 THEN PRINT "YOU DON'T NEE
D THAT": GOTO 10
9070 IF C24%=1 THEN 4115
9080 IF C15%<2 THEN 16050
9090 C24%=1:C15%=C15%-2:PRINT "HERE IS T
HE MAP":GOTO 10
9100 IF NM<>6 THEN PRINT "I HAVE NOT THA
T THING":GOTO 10
9110 IF C29%=1 THEN 4115
9120 C29%=1:PRINT "HERE IS YOUR RESIDENC
E-PERMIT.":PRINT "THAT'S FREE.":GOTO 10
9500 IF C>16 AND C<31 AND C<>23 THEN IF
C33%=1 THEN PRINT "A L'INTERIEUR D'UNE P
IECE,":PRINT "ON NE PEUT ALLER NULLE PAR
T":GOTO 10 ELSE PRINT "INSIDE A ROOM THE
RE IS NO WAY TO GO":GOTO 10
9510 IF C<17 OR C>30 THEN IF C33%=1 THEN
 PRINT "DANS QUELLE DIRECTION ?":GOTO 10
 ELSE PRINT "WHICH DIRECTION ?":GOTO 10
9520 IF NM<29 OR NM>30 THEN PRINT "DE QU
EL COTE ?":GOTO 10
9530 IF NM=30 THEN IM$="CIM23G":GOSUB 64
100:PRINT "CE SONT LES GARDES DU CARDINA
L.":PRINT "ILS SONT BIEN ARMES !":GOTO 1
0
9535 IF C16%=1 AND C18%=0 THEN PRINT "PA
S SANS PAYER LES CONSOMMATIONS":GOTO 10
9540 IF C16%=1 THEN PRINT "IL N'Y A PLUS
 PERSONNE":GOTO 10
9550 IM$="CIM23D":GOSUB 64100:PRINT "CE
SONT ATHOS, PORTOS ET ARAMIS.":PRINT "AP
RES EXPLICATIONS,":PRINT "ILS DECIDENT D
E VOUS SUIVRE":C16%=1:GOTO 10
10000 IF NM=0 THEN 4210
10010 IF C=21 THEN 10030 ELSE IF C=22 TH
EN 10090 ELSE IF C=32 THEN 10170 ELSE IF
 C=38 THEN 10200
10020 GOTO 4010
```

```
10030 IF NM<>7 AND NM<32 THEN IF VB=8 TH
EN 6550 ELSE 4100
10040 IF C11%=0 THEN 6640
10050 IF C11%>4 THEN PRINT "CE N'EST PAS
 LA PEINE : VOTRE CHEVAL ESTEN PARFAIT E
TAT":GOTO 10
10060 IF C11%=1 THEN C19%=1:GOTO 10070 E
LSE IF C11%=2 THEN C20%=1:GOTO 10070 ELS
E IF C11%=3 THEN C21%=1:GOTO 10070 ELSE
C22%=1
10070 C11%=0:IF VB=8 THEN PRINT "C'EST F
AIT : VOTRE CHEVAL A REINTEGRE  SA STAL
LE":FOR I=1 TO 2000:NEXT I:GOTO 1000
10080 PRINT "VOUS POUVEZ PRENDRE UN AUTR
E CHEVAL":FOR I=1 TO 2000:NEXT I:GOTO 10
00
10090 IF NM<>26 AND NM<>9 AND NM<>28 THE
N PRINT "ICI ON NE CHANGE QUE DU BON ARG
ENT OU  DES OBJETS DE VALEUR":GOTO 10
10100 IF NM=26 AND C5%=0 OR ((NM=9 OR NM
=28) AND C13%=0) THEN 6640
10110 IF NM=26 AND C5%=2 THEN PRINT "JE
VOUS LES AI DEJA CHANGE CONTRE DES  PIS
TOLES":GOTO 10
10120 IF NM=26 THEN C5%=2:C13%=900:PRINT
 "VOICI 900 PISTOLES EN ECHANGE":GOTO 10
10140 IF C14%=1 THEN PRINT "ON NE PEUT C
HANGER QU'UNE SEULE FOIS":GOTO 10
10150 IF C13%<450 THEN 16060
10160 PRINT "VOICI 45 LIVRES ANGLAISES":
C14%=1:C15%=45:C13%=450:GOTO 10
10170 IF NM<>7 THEN 4105
10180 IF C11%=6 THEN PRINT "YOU HAVE ALR
EADY CHANGED YOUR HORSE":GOTO 10
10190 C11%=6:GOSUB 64000:PRINT "OK, YOU'
VE GOT A NEW HORSE":GOTO 10
10200 IF NM<>7 AND NM<32 THEN 4100 ELSE
11020
10500 IF C<>38 THEN IF C33%=1 THEN PRINT
 "ICI VOUS N'ETES PAS CHEZ UN APOTHICAIR
E":GOTO 10 ELSE PRINT "IT'S NOT A APOTHE
CARY'S":GOTO 10
```

```
10510 IF C11%<>1 THEN PRINT "POURQUOI ?
IL EST EN BONNE SANTE":GOTO 10
10520 IF C13%=0 THEN 16080
10530 IF C13%<200 THEN 16060
10540 C11%=5:C13%=C13%-200:PRINT "IL A B
U DE LA POTION MAGIQUE. CELA VOUS COUTE
200 PISTOLES":GOTO 10
11000 IF C<>38 THEN IF C33%=1 THEN PRINT
 "VOUS N'ETES PAS CHEZ LE MARECHAL-FERRA
ND":GOTO 10 ELSE PRINT "IT'S NOT A BLACK
SMITH'S":GOTO 10
11010 IF C11%<>4 THEN PRINT "POURQUOI ?
IL EST BIEN FERRE":GOTO 10
11020 IF C13%=0 THEN 16080
11030 IF C13%<200 THEN 16060
11040 C11%=5:C13%=C13%-200:IF VB=15 THEN
 PRINT "VOICI UNE OCCASION PRESQUE NEUVE
";:ELSE PRINT "VOILA : IL A DES PNEUX NE
UFS";
11050 PRINT "CELA VOUS COUTE 200 PISTOLE
S":GOTO 10
11500 IF C13%=0 AND C33%=1 THEN 16080
11510 IF C15%=0 AND C33%=0 THEN 16070
11520 IF C=1 OR C=17 THEN NM=9:GOTO 6600
 ELSE IF C=23 THEN 11570 ELSE IF C=34 OR
 C=35 THEN 1500
11530 IF C=26 AND C28%=1 OR C25%=29 AND
C24%=1 OR C=37 AND C=3 THEN PRINT "THAT'
S DONE":GOTO 10
11540 IF C=26 AND C28%=0 OR C=29 AND C24
%=0 OR C=37 AND C36%<3 THEN PRINT "FOR W
HAT THING ?":GOTO 10
11550 IF C=38 AND C11%=5  THEN PRINT "C'
EST FAIT":GOTO 10 ELSE IF C=38 THEN PRIN
T "POURQUOI FAIRE ?":GOTO 10
11560 GOTO 4010
11570 IF C18%=1 THEN PRINT "VOUS AVEZ DE
JA PAYE":GOTO 10
11580 IF C17%=0 THEN PRINT "C'EST AU PAT
RON QU'IL FAUT PAYER":GOTO 10 ELSE IF C1
3%<50 THEN 16060
11590 C13%=C13%-50:C18%=1:PRINT "LE COMP
```

```
TE Y EST... MERCI":GOTO 10
12000 IF C=23 THEN 12020 ELSE IF C=34 OR
 C=35 THEN 12050
12010 IF C33%=1 THEN PRINT "CE N'EST PAS
 LA PEINE D'APPELER" ELSE PRINT "DON'T C
ALL, YOU LOSE TIME":GOTO 10
12020 IF NM<>31 THEN PRINT "PERSONNE N'E
NTEND":GOTO 10
12030 IF C16%=0 OR C18%=1 THEN PRINT "LE
 PATRON N'A PAS LE TEMPS":GOTO 10
12040 IM$="CIM23DP":GOSUB 64100:PRINT "V
OS AMIS ONT BU POUR 50 PISTOLES":C17%=1:
GOTO 10
12050 IF NM<>12 THEN IF C=34 THEN PRINT
"CHUT ! LES CALAISIENS FONT LA SIESTE !"
:GOTO 10 ELSE PRINT "HUSH ! THE CITIZENS
 HAVE A NAP !":GOTO 10
12060 C35%=1:IF C=34 THEN GOSUB 64000:PR
INT "BONJOUR, SI VOUS VOULEZ TRAVERSER":
PRINT "IL FAUT PAYER 300 PISTOLES !":GOT
O 10 ELSE PRINT "IF YOU WANT TO CROSS TH
E CHANNEL":PRINT "YOU MUST PAY 30 POUNDS
 !":GOTO 10
12500 IF C=19 THEN 12520 ELSE IF C=20 TH
EN 12540
12510 GOTO 4010
12520 IF NM<>11 THEN PRINT "DITES DONC,
MOUSQUETAIRE":PRINT "VOUS ETES TRES IMPO
LI !":GOTO 10
12530 C31%=1:C32%=1:PRINT "IL Y VA DE MA
 VIE !":PRINT "ALLEZ DEMANDER AU DUC DE
BUCKINGHAM":PRINT "DE VOUS REMETTRE LES
FERRETS."
12535 IF INKEY$="" THEN 12535 ELSE PRINT
 "VOUS AVEZ 3 JOURS POUR REUSSIR. VOICI
  UNE BAGUE QUI VOUS FERA RECONNAITRE.":
GOTO 10
12540 IF NM<>11 THEN PRINT "YOU HAVE BET
TER TO DO TO BE POLITE !":GOTO 10
12550 C31%=1:PRINT "GOOD MORNING, MONSIE
UR !":PRINT "WHAT CAN I DO FOR YOU ?":GO
TO 10
```

```
13000 IF NM=0 THEN 4210
13010 IF C=20 THEN 13040 ELSE IF C=31 TH
EN 13060 ELSE IF C=37 THEN 13080
13020 GOTO 4010
13040 IF C31%=0 THEN PRINT "YOU ARE IMPO
LITE, MONSIEUR !":GOTO 10 ELSE IF NM<>16
 THEN PRINT "HAVE YOU AN OTHER THING TO
SHOW ?":GOTO 10
13050 IF C32%=0 THEN 6645 ELSE C23%=1:PR
INT "HAVE YOU SOMETHING TO ASK ME ?":GOT
O 10
13060 IF C12%=0 AND NM<>6 OR C12%=1 AND
NM<>24 THEN PRINT "CELA NE VOUS AUTORISE
 PAS A ENTRER":GOTO 10
13070 IF C12%=0 AND C2%=1 OR C12%=1 AND
C3%=1 THEN PRINT "VOUS POUVEZ PASSER":C3
7%=1:GOTO 10 ELSE PRINT "VOUS N'AVEZ PAS
 CE QU'IL FAUT":GOTO 10
13080 IF C29%=0 THEN PRINT "YOU HAVE NOT
 THIS PAPER... TOO LATE !":GOTO 50000 EL
SE PRINT "THANK YOU, SIR !":GOTO 10
13500 IF NM=0 THEN 4210 ELSE IF NM<>14 T
HEN IF C31%=1 THEN 4100 ELSE 4105
13510 IF C=3 THEN 13530 ELSE IF C=4 THEN
 13550
13520 GOTO 4010
13530 IF C12%=1 AND C0%>220 THEN PRINT "
TROP TARD ! LA REINE EST EN PRISON ET...
VOUS ALLEZ LA REJOINDRE...":GOTO 50000
13540 C30%=1:PRINT "ENTREZ":GOTO 10
13550 IF C38%=0 THEN PRINT "THE DUKE HAS
 A MEETING, DON'T DISTURB":GOTO 10
13560 C30%=1:PRINT "COME IN":GOTO 10
14000 IF NM=0 THEN 4210 ELSE IF C<>17 TH
EN 14020 ELSE IF NM=6 AND C1%=0 THEN 413
0
14010 IF NM=6 THEN PRINT "Mr D'ARTAGNAN,
 IL EST DU PLUS HAUT INTE-RET QUE VOUS V
ENIEZ ME RENCONTRER RAPI- DEMENT. SIGNE
: VOTRE REINE":GOTO 10 ELSE GOTO 14060
14020 IF C<>33 AND C<>38 THEN 4010 ELSE
IF NM<>27 THEN 14060
```

```
14050 IF C=33 THEN PRINT "LA JUMENT BLEU
E :":PRINT "CLINIQUE VETERINAIRE, AU SUD
":GOTO 10 ELSE PRINT "ICI, ON PEUT REPAR
ER UN CHEVAL.":GOTO 10
14060 PRINT "CE N'EST PAS INTERRESSANT":
GOTO 10
14500 IF C=33 AND NM<>8 THEN 4100
14510 IF C=36 AND NM<>8 THEN 4105
14520 IF C=33 THEN C36%=1:GOSUB 64000:PR
INT "QUEL BEAU BOUQUET VOUS AVEZ !":GOTO
 10
14530 IF C=36 THEN C36%=2:GOSUB 64000:PR
INT "HERE IS A PRETTY BOUQUET !":GOTO 10
14540 GOTO 4010
15000 IF C<>26 THEN IF C33%=1 THEN PRINT
 "CE N'EST PAS UN RESTAURANT, ICI":GOTO
10 ELSE PRINT "YOU ARE NOT IN A RESTAURA
NT, HERE":GOTO 10
15010 IF C15%<10 THEN 16050
15020 C15%=C15%-10:C28%=1:PRINT "AH ! YO
U FEEL BETTER":PRINT "BUT YOUR PURSE LOS
T 10 POUNDS":GOTO 10
15500 IF C<>28 THEN IF C33%=1 THEN PRINT
 "CE N'EST NI LE LIEU, NI LE MOMENT":GOT
O 10 ELSE PRINT "IT'S NEITHER THE PLACE
OR THE TIME":GOTO 10
15510 IF C7%<>1 THEN PRINT "YOU ARE NOT
PROTECTED AGAINST COLD. YOU GET A SORE T
HROAT... TOO LATE !":GOTO 50000
15520 C27%=1:PRINT "EVERYBODY IS REST NO
W":GOTO 10
16000 IF C<>29 AND C<>37 THEN 4010
16010 IF NM=0 THEN 4210 ELSE IF C15%=0 T
HEN 16070
16020 IF C=37 THEN 16030 ELSE IF NM<>17
THEN 16040 ELSE IF C15%<2 THEN 16050 ELS
E C15%=C15%-2:C24%=1:PRINT "HERE IS YOUR
 MAP. IT'S 2 POUNDS":GOTO 10
16030 IF NM <>8 THEN 16040 ELSE IF C15%<
5 THEN 16050 ELSE C15%=C15%-5:C36%=3:PRI
NT "HERE IS A PRETTY BOUQUET FOR 5 POUND
S":GOSUB 64000:GOTO 10
```

```
16040 PRINT "NO ! I DON'T SELL YOU THAT"
:GOTO 10
16050 PRINT "YOU HAVE NOT ENOUGH MONEY":
GOTO 10
16060 PRINT "VOUS N'AVEZ PLUS ASSEZ D'AR
GENT":GOTO 10
16070 PRINT "NO ENGLISH MONEY IN YOUR PU
RSE":GOTO 10
16080 PRINT "VOTRE BOURSE EST VIDE":GOTO
 10
16500 IF C<>18 THEN IF C33%=1 THEN PRINT
 "ALLONS, SOYONS SERIEUX !": GOTO 10 ELS
E PRINT "WELL, DON'T GIVE WAY TO LAUGHTE
R !":GOTO 10
16510 IM$="C1M18P":GOSUB 64100:PRINT "EL
LE VOUS ENTRAINE DANS SA CHAMBRE...":PRI
NT "VOUS PASSEZ UNE FOLLE NUIT D'AMOUR..
.":PRINT "...TROP TARD !":GOTO 50000
17000 GOSUB 17700:IF V8=34 THEN 17500
17300 OPENOUT "J":PRINT #9,EN:PRINT #9,C
0%,C1%,C2%,C3%,C4%,C5%,C6%,C7%,C8%,C9%,C
10%,C11%,C12%,C13%,C14%,C15%,C16%,C17%,C
18%,C19%,C20%,C21%,C22%,C23%,C24%,C,C26%
,C27%,C28%,C29%,C30%,C31%,C32%,C33%,C34%
,C35%,C36%,C37%,C38%:CLOSEOUT:GOTO 10
17500 OPENIN "J":INPUT #9,EN:INPUT #9,C0
%,C1%,C2%,C3%,C4%,C5%,C6%,C7%,C8%,C9%,C1
0%,C11%,C12%,C13%,C14%,C15%,C16%,C17%,C1
8%,C19%,C20%,C21%,C22%,C23%,C24%,C,C26%,
C27%,C28%,C29%,C30%,C31%,C32%,C33%,C34%,
C35%,C36%,C37%,C38%:CLOSEIN:GOTO 2000
17700 CLS:PRINT "METTEZ FACE 1":WHILE IN
KEY$="":WEND:CLS:LOAD "D":IF PEEK(65529)
<>1 THEN 17700 ELSE RETURN
45000 RESTORE 45010:ENV 1,1,0,15,2,-7,2:
ENV 2,1,0,15,3,-1,1:FOR I=1 TO 18:READ F
1,D,E,F2:SOUND 17,F1,D,15,E,,1:SOUND 10,
F2,0.8*D,3,2*E:NEXT:RETURN
45010 DATA 47,28,1,56,47,28,1,56,47,28,1
,56,80,56,0,95,47,56,0,56,45,28,0,53,47,
28,1,56,47,56,0,56,53,28,1,63,53,28,1,63
,53,28,1,63,53,28,1,63,80,56,0,95,53,56,
```

```
0,63,47,28,0,56,53,28,1,63,53,56,0,63,60
,28,0,71
50000 RUN "FIN.BAS"
64000 IM$=STR$(C):IM$="CIM"+RIGHT$(IM$,L
EN(IM$)-1)
64010 IF C=17 THEN IM$=IM$+RIGHT$(STR$(C
10%),1)+RIGHT$(STR$(C4%+2*C6%+3*C8%),1):
ELSE IF C=21 AND C11%<>0 THEN R$=STR$(C1
1%):IM$="CCH"+RIGHT$(R$,LEN(R$)-1)
64020 IF (C=32 AND C11%=6) OR ((C=34 OR
C=35) AND C35%=1) OR (C=33 AND C36%=1) O
R (C=36 AND C36%=2) OR (C=37 AND C36%=3)
 THEN IM$=IM$+"P"
64100 LOAD "D":I=PEEK(65529)
64110 IF (C=1 OR C=5 OR C=6 OR C=7 OR C=
17 OR C=21 OR C=22 OR C=23) AND I<>2 THE
N I=2:GOTO 64400:ELSE IF (C=2 OR C=3 OR
C=8 OR C=9 OR C=15 OR C=18 OR C=19 OR C=
24 OR C=31 OR C=33 OR C=34 OR C=38) AND
I<>3 THEN I=3:GOTO 64400
64130 IF (C=4 OR C=10 OR C=11 OR C=12 OR
 C=13 OR C=14 OR C=16 OR C=20 OR C=26 OR
 C=27 OR C=28 OR C=29 OR C=30 OR C=32 OR
 C=35 OR C=36 OR C=37) AND I<>4 THEN I=4
:GOTO 64400:ELSE LOAD IM$
64200 INK 0,0:INK 1,0:INK 2,0:INK 3,0:CA
LL 34381:CALL 42589:INK 0,PEEK(65530):IN
K 1,PEEK(65531):INK 2,PEEK(65532):INK 3,
PEEK(65533):BORDER PEEK(65529):WINDOW #0
,1,40,22,25:CLS:RETURN
64400 CLS:PRINT "METTEZ LA FACE :";I:WHI
LE INKEY$="":WEND:CLS:GOTO 64100
```

PROGRAMME FIN

```
40000 IF PEEK(65528)=1 THEN LOAD "CIM39"
:INK 0,0:INK 1,0:INK 2,0:INK 3,0:CALL 34
381:CALL 42589:BORDER PEEK(65529):INK 0,
PEEK(65530):INK 1,PEEK(65531):INK 2,PEEK
(65532):INK 3,PEEK(65533):RESTORE 40040:
E=0:ENV 1,3,-5,8:ENV 2,1,-15,0:ELSE 5001
0
40010 FOR I=1 TO 2
40015 READ F1,D1,E1,F2,D2:IF F1=-1 THEN
40030 ELSE IF F1=-2 THEN I=2:E=1:GOTO 40
030
40020 IF INKEY$<>"" THEN END ELSE SOUND
1,F1,D1,15,E1:SOUND 2,F2,D2,4:GOTO 40015
40030 RESTORE 40040:NEXT:IF F1=-2 THEN 4
0010 ELSE IF E=1 THEN END ELSE RESTORE 4
0060:GOTO 40010
40040 DATA 24,14,0,47,14,24,14,0,40,14,2
4,14,0,47,14,40,14,0,53,14,30,14,1,60,14
,0,14,2,80,14,20,14,1,60,14,0,14,2,47,14
,20,14,1,40,14,22,14,1,45,14,24,14,1,47,
14,27,14,1,53,14,24,14,1,47,14,0,14,2,40
,14,30,14,1,47,14,0,14,2,80,14,24,14,0,4
7,14,24,14,0
40050 DATA 80,14,24,14,0,63,14,40,14,0,5
3,14,30,14,1,60,14,0,14,2,80,14,20,14,1,
60,14,0,14,2,47,14,20,14,1,40,14,22,14,1
,80,14,24,14,1,63,14,27,14,1,53,14,30,56
,1,60,56,-1,1,1,1,1
40060 DATA 32,14,0,63,14,30,14,0,60,14,2
7,14,1,53,14,0,14,2,63,14,24,14,1,80,14,
0,14,2,53,14,27,14,1,80,14,0,14,2,63,14,
24,14,1,80,14,0,14,2,63,14,27,14,1,53,14
,30,14,1,47,14,32,14,1,53,14,30,14,1,47,
14,27,14,1,53,14,0,14,2,60,14,32,14,0,32
,14,30,14,0,30
40070 DATA 14,27,28,1,27,28,32,14,0,63,1
4,30,14,0,60,14,27,28,1,53,28,20,28,1,40
,28,21,14,1,45,14,20,14,1,53,14,18,14,1,
```

```
60,14,21,14,1,71,14,20,6,0,80,6,18,6,0,7
1,6,20,6,0,80,6,18,6,0,71,6,20,6,0,80,6,
18,6,0,71,6,20,6,0,80,6,18,6,0,71,6,20,8
,0,80,8
40080 DATA -2,1,1,1,1
50010 ENT 1,150,1,1:ENT 2,20,-6,1,20,6,1
,20,-6,1,20,6,1
50020 SOUND 1,60,150,7,,1,1:SOUND 2,67,1
50,7,,1:SOUND 1,210,150,7,,1,1:SOUND 2,2
17,150,7,,1:SOUND 1,360,100,7,,2,1:SOUND
 2,367,100,7,,2
50030 I=0:WHILE I<1100:I=I+1:WEND: PRINT
 "VOULEZ-VOUS REJOUER (O/N) ?"
50040 R$=INKEY$:IF R$="" THEN 50040
50050 IF R$<>"O" THEN CLS:PRINT "AU REVO
IR":END
50060 CLS:PRINT "METTEZ FACE 1":WHILE IN
KEY$="":WEND:LOAD "D":IF PEEK(65529)<>1
THEN 50060
50070 LOAD "CIM0":INK 0,0:INK 1,0:INK 2,
0:INK 3,0:CALL 34381:CALL 42589:BORDER P
EEK(65529):INK 0,PEEK(65530):INK 1,PEEK(
65531):INK 2,PEEK(65532):INK 3,PEEK(6553
3):RUN "AVENT.BAS"
```

POUR UN CATALOGUE COMPLET DE NOS PUBLICATIONS

FRANCE
6-8, Impasse du Curé
75881 PARIS CEDEX 18
Tél. : (1) 42.03.95.95
Télex : 211801

U.S.A.
2021 Challenger Drive
NBR 100
Alameda, CA 94501

ALLEMAGNE
Vogelsanger. WEG 111
4000 Düsseldorf 30
Postfach N° 30.09.61
Tel. : (0211) 61.80.2-0
Telex : 08588163

Paris • Alameda • Düsseldorf

Les textes de ce livre
ont été saisis et mis en page
dans un environnement micro-informatique
puis tirés en électrocopie laser

par :

DESK

40, rue Léon-Blum
53000 LAVAL

Tél. (16) 43 68 13 67

Achevé d'imprimer le 4 décembre 1986 sur les presses de l'Imprimerie «La Source d'Or»
63200 Marsat - Dépôt légal : 4e trimestre 1986 - Imprimeur n° 2177

Ce livre vous guide pas à pas dans la réalisation d'un jeu d'aventure.

Vous trouverez, en première partie, les constituants d'un tel jeu, puis la mise en place du scénario, du plan et des conditions, suivie de la réalisation du programme BASIC. Pour finir, le tout sera agrémenté d'une partie son, largement commentée.
Tous les exemples fonctionnent sur les Amstrad CPC 464, 664 et 6128.

Un jeu d'aventure nécessite bien sûr l'utilisation d'images graphiques en grand nombre. Pour vous faciliter la tâche, vous trouverez, en seconde partie, la description complète d'un logiciel d'aide à la création d'images doté d'un compresseur/décompresseur d'images graphiques, qui vous permettra un gain de place très important pour le stockage de vos images sur disquette.

Ce livre se termine par le listing complet d'un jeu d'aventure dont la solution n'est pas explicitée.

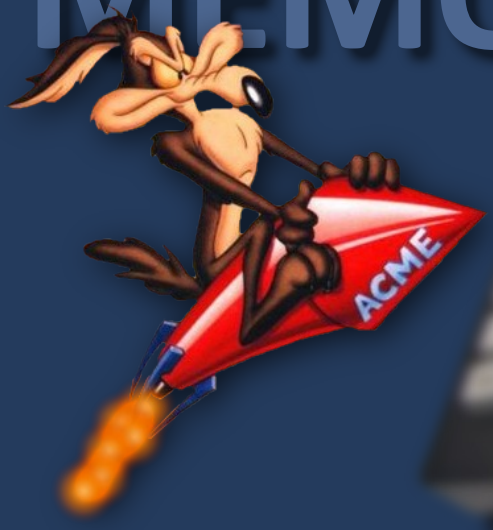
0204 1286 128 F

9 782736 102043

Document numérisé avec amour par :

AMSTRAD CPC

MÉMOIRE ÉCRITE



https://acpc.me/