Programmez votre CPC ou CPC+ avec

numéro 18

QUASAR

JS

- ACTUS
  - ~les meetings
- TESTS
  - ~Fres Fighter
- ASSEMBLEUR
  - ~soft : compression
  - ~hard : interruptions
  - ~coding : réseau
- DOSSIER
  - ~programmez vos jeux
- LANGAGE C
  - ~initiation au Small C

numéro 18

QUASAR

-sommaire ——— pages-

EDITORIAL

Ouf ! Le numéro 18 de Quasar CPC est enfin chez vous ! Nous sommes désolés pour l'énorme retard de parution, mais les choses ont été très compliquées cette fois-ci encore. Enfin, nous espérons que celui-ci saura vous satisfaire, car il y a pas mal de nouveautés telles qu'un dossier sur la programmation des jeux, ou un cours sur le Langage C.

Et puis, cette fois-ci, c'est une équipe très complète qui a participé à l'élaboration du fanzine puisque nous étions six. De plus, Rémi VINCENT nous a fait le plaisir de nous donner quelques dessins originaux pour l'illustration de nos pages. En outre, le Quasar Disc 2 est également disponible dès à présent avec une nouvelle interview croustillante et un kit de développement C pour CPC. Aussi, que ce retard ne vous inquiète pas, Quasar CPC se porte mieux que jamais.

En ce qui concerne la période de parution, vu les circonstances actuelles, il nous est difficile de faire mieux, et nous en sommes vraiment désolés. D'aucuns voudraient bien un Quasar CPC mensuel de 30 pages - en fait nous aussi on adorerait vous proposer ça - mais, soyons réalistes, ce n'est pas possible. Quant à la PAO sur CPC, nous n'envisageons pas d'en changer. A force de déporter les activités CPCistes sur d'autres machines le CPC va devenir une pièce de musée dont on parle beaucoup mais qui ne sert plus à grand chose !

Il m'est d'avis qu'un ordinateur demeure réellement en vie s'il continue à être utilisé pour autre chose que la bidouille ; sinon, il est condamné à plus ou moins long terme. C'est la raison pour laquelle nous conservons un maximum d'activité sur le CPC : PAO de Quasar CPC (Oxford PAO et AMX Pagemaker), courrier de la rédac' (PROTEXT), scans (Scanner DART et VIDI-CPC), fichiers d'adresses (dBaseII), etc... Mais il est des domaines pour lesquels le CPC n'est pas à la hauteur, pour l'Internet par exemple, j'ai du me résigner à utiliser mon Amiga... mais j'ai bon espoir de pouvoir surfer avec mon CPC d'ici peu.

Aussi, je ne comprends pas ceux qui refusent toute évolution du CPC. Qu'en serait-il aujourd'hui de notre CPC si nous n'avions pas eu droit à toutes ces merveilleuses extensions que sont la Multiface II, le Scanner DART, le VIDI-CPC, le Romboard, la souris AMX... le lecteur 3"1/2 ! Leur existence ne remet pas en cause le développement sur CPC de base ; au contraire, elle y contribue ! Alors pourquoi cette levée de boucliers lorsqu'on parle de carte son 8 bits, de carte accélératrice, de modem, de disque dur ? Si le CPC permettait de manipuler confortablement les médias informatiques actuels, ne pensez-vous pas que cela lui serait bénéfique ?

OffseT

ACTUS

Enfin nous nous retrouvons pour de nouvelles actus en provenance du petit monde du CPC ! Cette fois-ci encore ces quelques pages seront bien garnies. Pour commencer je vous propose de faire le point sur les traditionnels meetings estivaux qui nous attendent.

Ze Meeting 2000

Bassoues et les CPCistes... Une longue histoire d'amour ! Pour la sixième fois tous les CPCistes sont invités au Ze Meeting ! Cette fois-ci encore, celui-ci se déroulera dans la charmante bourgade de Bassoues, dans le Gers profond. Une petite différence toutefois par rapport aux précédentes éditions puisque celui-ci n'est plus organisé par Nicky One mais JRM45. Donc, si vous n'avez toujours pas reçu d'invitation, une seule adresse :

VAUTARD Jérémie
3, impasse de la pépinière
45430 CHECY

Comme d'habitude, tout le monde est le bienvenu du moment qu'il ne vient pas pour échanger ses dernières compil' MP3 ! Les dates : 4, 5 et 6 Août.

Warf Meeting

Il s'agit là aussi d'un rendez-vous annuel ! Ce meeting est toujours organisé par Nik'ro, le rédacteur en chef de notre confrère le Fanss et devrait avoir lieu lors du dernier week-enk de Juillet à Montluçon. Il y a pour l'instant des problèmes de réservation de salle.

Vous pouvez vous renseigner sur le site roudoudou.com ou en écrivant à :

Mickael FOUCAUX
41, rue du champ du Paradis
03100 MONTLUÇON

L'aRts 2000

Contrairement aux deux précédents, il ne s'agit pas là d'un meeting spécifique au CPC mais d'une coding party spécialement dédiée à toutes les vieilles machines (CPC, C64, MSX, TO7, Atari ST, Amiga A500). Bien sûr, il y aura aussi des machines plus récentes (Amiga, Falcon, Mac, PC, etc...) mais les CPCistes y sont prestement invités ! L'an passé il y avait seulement quelques C64 et pas de CPC ! Alors on compte sur vous corriger ça cette année... Ce meeting aura lieu le premier week-end de Septembre à Grenoble. Je n'ai pas plus de détails pour le moment mais le site de la party devrait bientôt être mis à jour :

http://www.rtsy.org

Les fanzines

Eh bien c'est triste à dire, mais depuis le dernier numéro de Quasar CPC tout est resté assez calme de ce côté là. Il faut toutefois souligner la sortie du numéro 10 d'Amslive, fin Juin, et féliciter une fois de plus les frères Thomasette pour leur fanzine mensuel toujours au rendez-vous et jamais en retard, Another World. Je ne vais pas détailler dans ces pages le contenu de ces fanzines que je vous invite à commander si vous n'en êtes pas des lecteurs assidus :

ANOTHER WORLD
Thomasette Franck & David
26, rue des Maisonnettes
54300 LUNEVILLE

AMSTRAD Live
amslive@mail.dotcom.fr
www.multimania.com/amslive

Dans un registre plus comique, je me dois également de citer les quelques numéros du "Drapeau Noir" qui ont innondé nos boîtes aux lettres ces derniers temps. Bien joué, ça en a fait cogiter plus d'un !

Les nouvelles productions

Côté démos nous avons été plutôt gatés. Madram nous a enfin présenté sa production : Ecole Buissonière. Il s'agit là de l'une des meilleures demos soft pour CPC. Je crois que l'on peu dire que cette démo a fait autant de bruit que l'Ultimate Megademo de Face Huger en son temps. En effet, quasiment tous les effets sont complètement inédits (plasma soft, roto-zoom, etc...) et valent vraiment le détour : c'est fluide, c'est propre, c'est bien pensé. En bref, du grand art. Un petit reproche toutefois, les couleurs, si elles rendent bien sur CPC ancienne génération, auraient pu être travaillés pour les CPC+. Je n'en dirai pas plus, si vous ne l'avez pas encore vu, demandez-là à la rédac'.

Je ne vais pas développer ici les autres actus concernant les dernières productions en date car je pense que tout a été dit, ou presque, dans les derniers numéros d'Another World. De plus, étant donné que nous ne pouvons pas assurer de parution régulière en ce moment nous nous contenterons des news les plus importantes ou qui sont spécifiques à nos propres productions. Je vais donc enchaîner par ce qui se passe chez nous.

Ce que nous préparons

L'Asic Inside, la demos specifique aux CPC+ qui vous est promise depuis des années est en passe d'être achevée et nous avons bon espoir que vous la présenter lors de Ze Meeting 2000... Oui, je sais, on dit la même chose tous les ans, mais cette fois-ci, c'est la bonne !

Pourquoi en sommes-nous convaincus ? Eh bien car le fameux Soundtracker DMA indispensable à la finalisation de cette demo, est déjà parfaitement fonctionnel ! Et la première musique DMA de Zik sera celle de Stormlord+ que SNN doit achever avant son départ pour les Etats-Unis... He, mine de rien, en deux paragraphes, je viens de vous annoncer la sortie de 3 productions pour cet été !

Pour parler du présent, sachez que vous pourrez retrouver une nouvelle ROM pour vos Ramcards sur le Quasar Disc 2. Il s'agit du Citron Boot. Une ROM permettant de reconfigurer à souhait le boot de votre CPC. Il ne s'agit pour le moment que d'une version beta dans laquelle de paramétrage est peu pratique, mais la plupart des fonctions sont là (choix du mode et des couleurs, affichage ou non d'images de boot, etc...).

Je suis également en train de travailler sur le Citron Viewer, un utilitaire qui permettra de visualiser différents types d'images (IFF, GIF, BMP, PI1, ...) avec des modes graphiques les plus adaptés possibles (overscan, rasters). Il sera bien sûr ensuite possible de convertir ces images au format CPC.

Attention, ne vous fiez pas aux noms, ces logiciels n'ont rien à voir avec la Citron Demo présentée lors du Ze Meeting'99 et qui en a fait rire jaune plus d'un !

Du matériel neuf pour CPC

Nous sommes désormais très nombreux à être passé au format de disquettes 3"1/2... Mais un problème se pose quant à la compatiblité des lecteurs actuellement en vente chez les épiciers. Nous avions rédigé un article complet sur la question dans notre numéro 14 mais il se trouve qu'une solution plus simple est possible. En effet, grâce à l'intervention de l'ums de nos lecteurs, nous savons maintenant qu'il existe encore des constructeurs fabricant des lecteurs 3"1/2 100% compatibles CPC. Ainsi, si vous êtes suffisament nombreux à être intéressés, nous pourrions tenter de vous proposer des lecteurs 3"1/2 prêts à l'emploi pour 300F environ.

Materiellement parlant, une nouvelle liste de diffusion concernant la conception d'un libre descendant du CPC a été ouverte il y a peu sur l'Internet. Pour plus de détails jetez un coup d'œil sur :

www.egroups.com/group/cpcng

Toujours là ?

Ces derniers temps, nombreux sont les anciens CPCistes à renouer des liens avec la scène d'aujourd'hui. Certains préparent même quelque chose de spécial pour leur retour ! C'est ainsi que Power System nous annonce quelque chose de spécial pour très bientôt, notamment sur l'Internet. Je vais également en profiter pour faire un peu de pub à Epsilon qui sort son premier CD 2 titres pop/dance. Si vous voulez lui donner un coup de pouce contactez-le :

Christophe KASTRIOTTIS
18, rue Pierre Curie
78700 CONFLANS SAINTE-HONORINE

power-systeme@infonie.fr (contact : ALN)

Un bug de plus !

Juste une petite brève pour vous signaler un bug du fameux Soundtrakker. Tony a mis le doigt dessus par hasard : lorsque vous enregistrez une musique avec un nom de fichier commençant par un chiffre, vous ne pourrez plus le charger !

L'interview

Pour finir les actus de ce numéro 18, je vous propose une interview de Tomy (Pierre Chibleur), qui va nous parler de la passion qui l'anime en dehors du CPC : l'esperanto.

Quasar CPC : Bonjour Pierre. Comment es-tu arrivé sur la scène CPC ?

Tomy : Au départ, j'ai acheté un 464 plus (oui, ça existe : avec Burnin' Rubber et des tas d'incompatibilités !). C'était en Août 1991. Mon but était d'apprendre l'informatique sans dépenser des fortunes.

Assez vite, et grâce à Patrick Aubert (Runstard) et Jean-Michel Cossart (Info Système CPC), j'ai découvert la richesse des 8 bits et, dans la foulée, acquis un 6128 de l'ancienne génération. Dans le même temps, l'affaissement commercial d'Amstrad se confirmait, mais les connaissances que j'accumulais peu à peu firent que j'ai cherché à me rendre utile. Je connaissais l'esperanto : j'ai allié mes deux passions, j'ai crée Tomy Esperanto/CPC International.

Pour moi, l'esperanto est une vraie solution : compréhensible rapidement, neutre par respect des langues d'origine de chacun, crée pour l'internationalité, s'il était enseigné, l'esperanto deviendrait ce pont entre tous les hommes rêvé par Louis-Lazarre Zamenhof, son créateur.

Pour me confirmer, des fanzines sortaient en plusieurs langues : Rundschlag (allemand, français, anglais, grec), Eurostard (français, anglais, espagnol). Quelle énergie dépensée !

Quasar CPC : Quelles sont les productions d'Esp/CPCI ?

Tomy : Au départ, c'était du Basic, genre PAK (programmes au kilo). Il y a eu les caractères spéciaux (CARESPO), la prononciation (PRONESPO). Puis, j'ai découvert et fait connaître le travail de Jean-Sébastien Brilleaud, avec Esperant, une présentation de la langue avec musique, graphisme, animation ludique. "Du très bon travail", en a dit Quasar CPC...

Et je m'y suis mis : à partir d'un programme d'ACPC, j'ai fait LEKSESPO, lexique performant que l'on peut étoffer soi-même. ROM a crée VOC dans Road Runner ; j'en ai fait VOCESPO, que je continue à améliorer en tenant compte des qualités lexicologiques de la langue.

Quasar CPC : Et maintenant ?

Tomy : Je suis passé à une phase plus créative. Grâce à une PAO simplifiée de P. Joniec, publié dans Micro-Mag, j'essaye de reproduire des œuvres littéraires dans le genre de la série des Ballades, très bien faites graphiquement. J'ai en ce moment PRINCETO en chantier, sur la traduction de Pierre Delaire intitulée "La eta princo" (Le petit prince, d'Antoine Saint Exupéry). Je crée des dessins avec OCP. J'essaye des animations de couleurs. Je vais sans doute travailler avec Outch!Writter, que mon ami Tom Pouce vient de me faire parvenir.

Je fais aussi, pour un camarade allemand avec qui je correspond en esperanto (encore merci Markus Buntru), de l'"espérantisation" de logiciels, comme l'Ecrivain par exemple. J'ai commencé à traduire des notices en langue internationale, mais c'est difficile : je vais faire des appels dans le monde esperantiste. Peut-être que cela donnera de nouveaux adeptes au CPC !

Quasar CPC : Justement, quelles relations entretiens-tu avec la scène ?

Tomy : Pour moi, faire du CPC, c'est de la création, tant au niveau de la programmation, que du point de vue de la qualité des réalisations.

Avec l'esperanto, j'estime que je peux rendre service à la société.

A la scène, je propose tout cela en bloc : à chacun de faire le tri, de prendre ce qui l'intéresse. J'ai déjà eu les très gratifiantes surprises de voir Serge Cayrier (alias Sirius, était le gestionnaire du CPC Club sur Minitel) se mettre à l'esperanto (salut, les Pays Roannais !), ainsi que ROM, qui a dessiné pour Laüte (bimestriel esperantiste), Grees m'a demandé de l'information...

Mon club Esperanto-Oyonnax utilise un 6128 plus pour ses jeunes. Il y a des esperantistes partout dans le monde, n'est-ce pas prometteur ?

Pour de plus amples informations, il existe Esperanto Info, très bien réalisé par Robert Llorens, de Nice. Ses coordonnées : Les Hortensias II, 121 bis, Bld Napoléon III, 06200 NICE, Tél : 04.93.83.08.73.

Quasar CPC : Une réflexion hautement philosophique ?

Tomy : Oyonnax vaincra !

Quasar CPC : Merci Tomy !

Voilà, c'en est fini des actus pour ce numéro 18, la prochaine fois vous aurez droit aux comptes rendus des meetings de cet été...

OffseT

PETITES ANNONCES

Vends, pour émulateur CPC sur PC, un CD AMSTRAD CPC : + de 2000 logiciels (jeux, demos, fanzines disc, utilitaires, slide show, éducatifs) au prix de 50F (port compris).
Vends également fanzines papier scannés en GIF et gravés sur 3 CDs pour 50F le CD (port compris). Cédric BAROU, 20 rue de la Charité 13150 TARASCON.

Vends matériel AMSTRAD CPC : clavier 6128, écran couleur, scanner DART, DMP2000, digitaliseur de voix, Multiface 2, souris, Romboard avec plusieurs ROMs... Pour tout renseignements (faire offre) : Frédéric MOLL, 9 Lotissement la Farigoule, 13150 TARASCON.

Vends (faire offre) :
- carte son compatible Sound Blaster (port ISA) avec discs installation. TBE.
- modem externe 28800. TBE.
- disque dur 428Mo de marque SEAGATE (modèle ST3491A).

Contacter Cédric BAROU, 2 Av. de la République 13150 TARASCON

Vends matériel et logiciels :
- Lecteur externe 3 pouces 200F
- Carte extension disc pour 464 100F
- Une dizaine de disquettes 5"1/4 100F
- Une dizaine de disquettes 3" 70F
- De la documentation sur l'assembleur 60F
- Disquette de cours asm (doc + disc) 50F
- Big flasher pour tout programmer ce que l'on veut à base RSX, doc + disc 60F
- Procopy+, disc et doc 50F
- 3D Construction KIT 100F
- Equinoxe (Soft de musique), disc + doc 60F
- Masterfile III, disc et doc 60F
- 2 cartouches CPC+ 40F pièce
- 2 barettes mémoire (32 bits) EDO 8Mo 100F
- 2 barettes mémoire (32 bits) EDO 4Mo 70F
- 1 clavier 102 touches 70F
- Graphic Works (dessin industriel) 50F
- Lecteur 5"1/4 (sans alimentation) 250F
- Logiciels Microsoft (Money 99, Works 4.5a, Atlas mondial encarta 99) avec licence, le tout pour 400F

Richard TREHET
e-mail : integra.@wanadoo.fr

Recherche imprimante et logiciel de traitement de texte pour Amstrad CPC 464.

Sophie DESPORT
e-mail : sdesport@club-internet.fr

Vends :
- Lecteur de disquettes 5"1/4 avec disquettes d'installations et vierges pour Amstrad CPC 6128 à 300F
- Imprimante DMP3000 avec ruband d'encre, nappe de connexion à 400F

Olivier PLICQUE
e-mail : voodoo32@caramail.com
Téléphone : 06 60 59 50 39

Vends ensemble ou séparément un Amstrad 6128 PLUS en parfait état avec une manette de jeu, la cartouche "Burning Rubber", des disquettes (jeux, utils, éducatifs) avec codes et plans nécessaires au fonctionnement de certains jeux. 25 disquettes sont en vente (dont le CP/M Plus), vous pouvez en demander la liste à la rédaction de Quasar.
Faire offre.

Alain GUYOMARD
e-mail : alain.guyomard@wanadoo.fr

Vends : * Utilitaires :
- Pascal / MT+ (Digital Research) 1D7+livre 99F
- CBasic Compilateur (Digit.Res) 1D7+livre 99F
- Smartkeys (Amsoft) 1D7+livre 29F
- Editeur/(Dés)assembleur (Amsoft) 1K7+doc 29F
- Gest. base de données (Amsoft) 1K7+livre 29F
- Decision Maker (Amsoft) 1K7+livre 29F
- Wordstar (trait.txt Micropro) 1D7+livre 59F
- Super Paint (Micro Application) 1D7+doc 79F

* Livres :
- La bible du graphisme (Micro Application) 39F
- La bible de l'assembleur (Micro Appl.) 29F

* Jeux :
- Dungeons&Dragons (US Gold) 2D7+livret 129F
- Space Ace 7 Compil (Star Games) 1D7+doc 139F (Cybernoid, Exolon, Northstar, Trantor, Venom Strike Back, Xevious, Zynaps)
- Les vainqueurs compil (US Gold) 2D7+doc 139F (Blasteroids, Forgotten Worlds, Last Duel, Tiger Road, Thunder Blade)
- Stormlord (Hewson) 1 disquette + livret 59F
- Les Passagers du Vent 2 (Glénat) 1D7+doc 59F

e-mail : Christophe.Vangyzegem@swing.be

Si le seul contact pour la PA qui vous intéresse est un e-mail et que vous n'y avez pas accès, écrivez à la rédaction qui fera suivre.

LA RUBRIQUE X

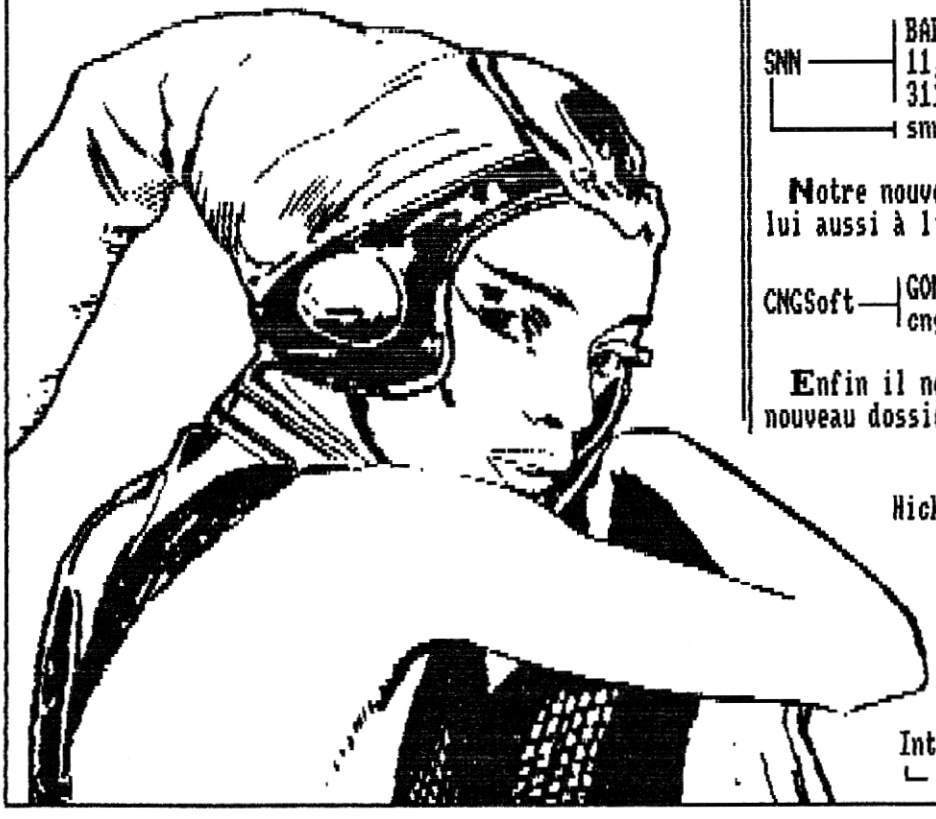
Six ! Oui, nous sommes cette fois six rédacteurs à avoir participé à ce numéro 18. J'espère que l'opération pourra se renoubeler car la diversité et la richesse des sujets abordés s'en trouvent améliorées...

Comme celà est désormais coutume, voici tout d'abord l'adresse de la rédac' :

La rédac' —
RIMAURO Philippe & Gilles
8, chemin des Maillos
09200 SAINT-GIRONS
quasarcpc@free.fr

Il s'agit là de l'adresse principale du fanzine à laquelle vous pouvez adresser tous vos courriers et plus particulièrement ceux concernant les abonnements et les commandes d'anciens numéros. Nous y attendons également vos réactions et vos conseils pour que notre fanzine puisse sans cesse progresser.

Ensuite, vous pouvez bien évidemment écrire directement aux rédacteurs de certains articles. Je vous donne donc tout d'abord les coordonnées de Zik, notre spécialiste en électronique :



Zik —
RIMAURO Gilles
6, allée des Sciences Appliquées
Résidence 7 - Appt 892
31400 TOULOUSE
grimauro@etud.insa-tlse.fr

Ensuite il y a Tony, de la section "Relations Publiques" que nous vous invitons à contacter pour tout ce qui concerne l'actualité du CPC :

Tony —
RENEAUT Antoine
Séminaire Saint Luc
7, cours de la Trinité
13625 AIX en PROVENCE Cedex 1
arenaut@wanadoo.fr

Fidèle parmi les fidèles, SNN est toujours présent dans nos pages et reste à votre écoute pour ses futurs articles sur la programmation du CPC(+) :

SNN —
BARDENAT Thomas
11, avenue Aignan Carrière
31120 LACROIX-FALGARDE
snn@mailgsm.com

Notre nouveau collaborateur espagnol, CNGSoft, est lui aussi à l'écoute :

CNGSoft —
GONZALES César Nicolás
cngsoft@lettera.net

Enfin il ne faut pas oublier Hicks qui a lancé un nouveau dossier sur la programmation des jeux :

Hicks —
THOMASETTE Franck et David
26, rue des Maisonnettes
54300 LUNEVILLE

Pour finir voici mon adresse :

OffseT —
RIMAURO Philippe
rimauro@ifrance.com

Internet : http://www.chez.com/futurs
└ ou bien http://www.ifrance.com/futurs

Histoires Perpendiculaires

"DESSINE-MOI UN MOUTON"

Vegapolis, 17h et quart, le rythme de la cité poursuivait son cours, les hommes menaient leurs affaires avec entrain, lorsque Théo, huit ans, quémanda : "Dessine-moi un mouton !"

Un enfant demandait qu'on lui dessine un mouton ! A Vegapolis, les enfants sont rois : mais comment faire pour que la demande de celui-ci soit exaucée ? Comme personne n'était arrivé à lui répondre, on réunit le Conseil, afin de débattre sur la meilleure manière de figurer le mouton. Plusieurs idées furent évoquées.

Les uns pensèrent que le mieux était de prendre les mesures de la bête, et en additionnant le quotient des largeurs, il en résulterait une équation qui donnerait la longueur de la forme quadréidale à tracer.

D'autres voyaient dans l'animal la figuration de la relation de l'homme à la nature, sa place prédominante dans le monde par rapport à sa bestialité : à cause de sa capacité à prendre conscience de cela, il fallait représenter un homme avec un grand cerveau.

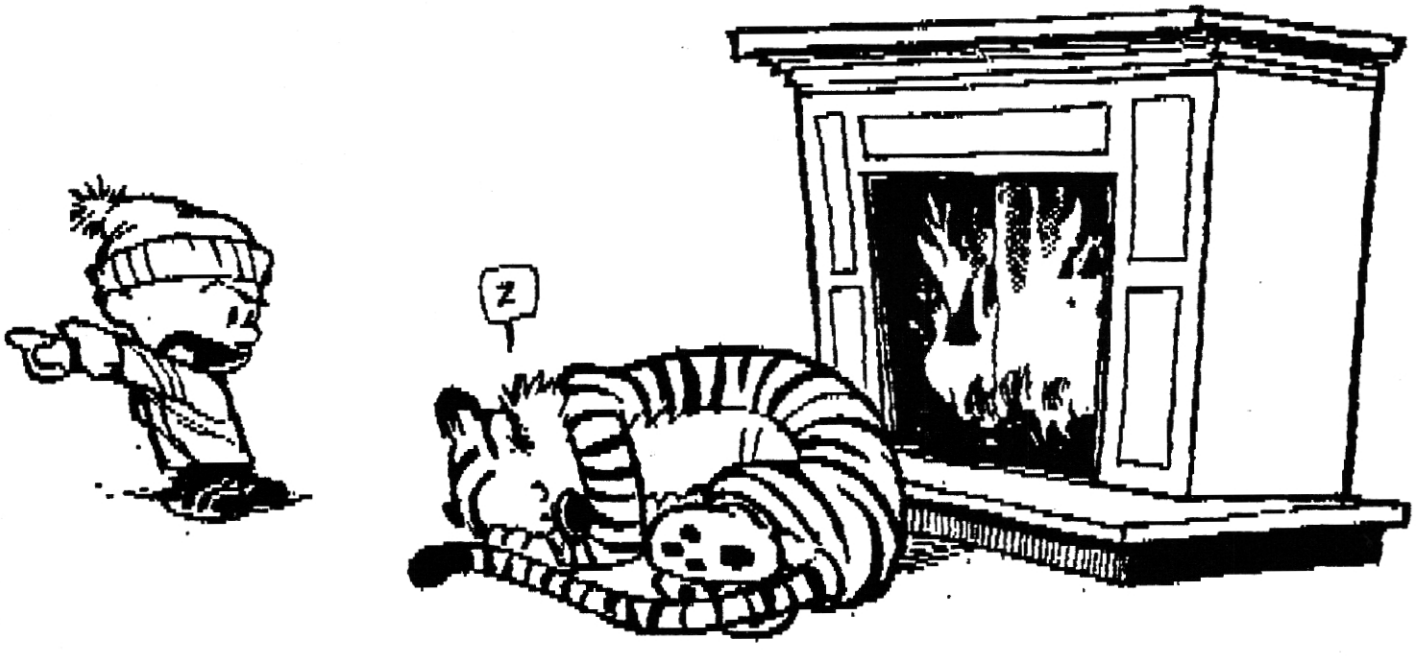
D'autres encore, proposèrent d'exposer le problème au Grand Assembleur, qui lui seul, par son aptitude au calcul magique, donnerait la réponse.

Face au doute, face à la peur de ne pas trouver de quelle manière dessiner un mouton, quelques-uns proposèrent que l'on aille toucher la Statuette, car les effets spectaculaires et l'envoutement qu'elle provoque auraient aidé l'auteur-élu dans son oeuvre.

Que de confusion à Vegapolis ! Au milieu de toutes ces propositions, comment satisfaire le besoin de cet enfant ? Qui écouter, quelle solution adopter ? Finalement, le chef de la métropole des grands-hommes décida que l'on ébauche un carré de quatre traits pour extraire de l'animal son être le plus profond.

Et Théo s'en alla, le coeur tout triste, parce-que les "grands" n'avaient pas su reconnaître dans sa demande un appel à redevenir comme lui, même l'espace d'un court instant...

Tony



TESTS

FRES FIGHTER II turbo

Qu'ils sont beaux nos petits amis !

En voilà un jeu qui mérite notre attention ! En effet ce jeu est... 1) tout récent (sorti en fin d'année 1999) 2) amusant (plutôt utile pour un jeu) 3) assez bien fait 4) vous allez arrêter avec vos questions embarrassantes ?!

Tout d'abord, le petit run"ff commence sur les chapeaux de roue avec une reconnaissance des caractéristiques de votre CPC, notamment les lecteurs de disquettes, ce qui est particulièrement sympathique, parce qu'avec les 2 disquettes du jeu, vous n'allez pas avoir besoin de changer sans arrêt de face... Ce qu'il y a d'interressant dans ce démarrage, c'est le copieur intégré (si vous arrivez à trouver le moment où appuyer sur les bonnes touches pour le mettre en marche !), même s'il a fallu dans mon cas le relancer à chaque fois pour les différentes faces ; de toute façon le seul moyen de copier le jeu est de passer par là, Mrs Discology ou Copyluck ne veulent pas entendre parler de ces disquettes au format d'une autre race que la leur. J'étais sûr que le CPC était raciste... et oui c'est comme ça...

Mais tout ça ne nous dit pas grand chose sur le jeu alors c'est sans plus tarder que j'en commence la présentation. Belle transition, puisque justement, dans une option du chargement, Bollaware nous explique que cette oeuvre est leur dernière après de nombreuses années de travail sur 9 jeux (le dernier en date était Blackland, cf. Quasar 17). Cette part' intitulée "Legacy of Bollaware" est originale est bien comique sur la fin. On est triste, mais ça commence bien.

Ensuite vient l'intro du jeu, qui en vaut vraiment la peine : tout à fait dans le style Nintendo ou anciennes bornes arcade, les personnages nous sont présentés de manière presque déifiée. Quelle beauté dans ces animations : flashs, trainée, mouvements parfaits et synchronisation musicale, un vrai spectacle ! L'intro est longue, encore une fois chargée de traits comiques (il faut dire que le jeu lui-même est une sorte de parodie des jeux de baston, cqfd dans le titre) et elle ne manque pas de vitalité.

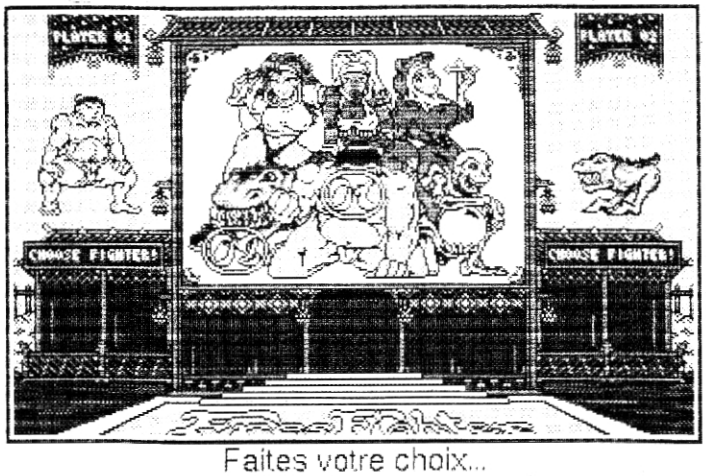
La barre de progression du loader m'indique que nous arrivons bientôt au menu. Je constate qu'en revenant au menu les fois suivantes, la musique n'est pas toujours la même : c'est agréable. J'en profite au passage pour dire que les zicmus ne sont peut-être pas fantastiques mais elles ont le mérite d'être parfaitement adaptées au style de jeu : elles entreront donc dans la catégorie des soundchips inoubliables (quand on entendra désormais ces musiques on se dira : "tiens, ça c'est FF2T !"). Pour ce qui est du menu, on s'y déplace lentement mais sûrement, et de nombreuses possibilités de jeu sont offertes. Vous y trouverez notamment :

- le nombre de rounds réglable de 2 à 7.
- l'activation / désactivation des effets spéciaux : la pluie, les éclairs, l'assombrissement des décors et donc de la zone de combat, et j'en passe car très vite j'ai été amené à les désactiver. Et oui, ils rendent très bien mais au détriment de la jouabilité. Pour un FF2T jouable, et surtout rapide, il faut malheureusement désactiver les effets spéciaux.
- 2 niveaux de difficulté : en "baby" c'est déjà assez difficile comme ça !
- le choix de l'arène : à chaque personnage correspond un environnement de combat (forêt, cimetière, église etc... mais pourquoi avoir choisi une bonne soeur comme un des 6 personnages ? quel humour bizarre ! (il fallait bien que le Tony fasse sa petite remarque !)).

Le menu principal

- Up to 16 : le jeu en tournoi. A tester lors de nos prochains meetings. Cette option peut s'avérer très interressante...
- Single Battle : pour choisir son adversaire.
- Arcade Mode : contre combien d'adversaires arriverez vous à combattre avant de perdre une vie ?
- Championship : celui qui mange le plus de chips a gagné, c'est lui le champion. A moins que ce ne soit comme dans Mortal Kombat, où le principe du jeu est de monter en niveau de difficulté : c'est le mode de jeu tout seul le plus riche.
- Survival : de combats en combats, votre barre d'énergie ne remonte pas. Tiendrez-vous jusqu'au bout où bien comprendrez-vous l'intérêt d'un bon médecin ?

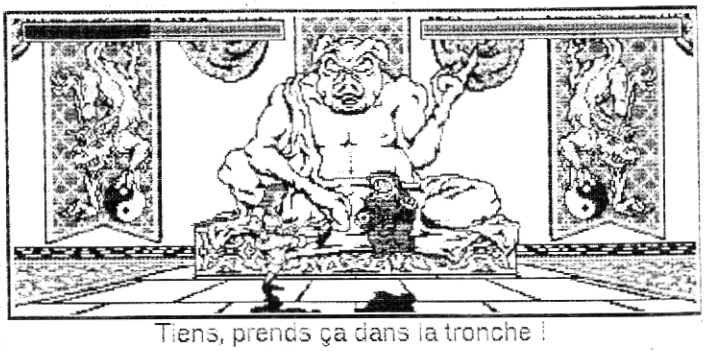
Au premier abord, ce jeu de baston m'avait semblé primitif, et mis-à-part le côté très "fun" de sa réalisation, il n'apportait rien à la logithèque du CPC. Je me trompais. Ce qui amène à réfuter ce jeu, ce sont toutes ces premières impressions qui sont bien loin de la réalité (c'est très métaphysique). Ces impressions, quelles sont-elles ? Tout d'abord le graphisme d'Uncle Siggi, caractéristique mode 1 couleurs bizarres, mais il faut rentrer dans cet art spécial. Ensuite, les bruitages et la musique de Tom & Jerry manquent de qualité, en plus, pendant les combats il n'y a pas de musique ; cependant, comme je l'ai déjà dit, les musiques ont une mélodie que l'on retient. Mais ce qui m'a repoussé au tout départ, c'est la lenteur des animations. Certes, les mouvements sont parfaits, excellemment pensés et dessinés, et pourtant, c'est fou d'en arriver là, mais il y en a trop ! Cela ralentit le rythme de l'action. Avec quelques Mhz de plus, le CPC aurait pu calculer rapidement tout cela, mais là



Faites votre choix...

c'est trop... Attendez ! N'arrêtez pas la lecture ici car ce n'étaient que des impressions ! En désactivant les effets spéciaux (pourtant si chouettes) vous obtiendrez un FF2T tout-à-fait jouable, et même très agréable ! Une fois que vous vous serez entrainé tout seul dans un des modes Arcade, Champion des Chips ou Survival, que vous aurez appris à maîtriser les coups spéciaux (enchaînements de touches / joystick), revenez à l'option Single Battle et jouez contre un véritable adversaire ! ça marche à 2 joysticks, ou au clavier, même si les touches vous sont imposées.

En conclusion, je vous invite à vous pencher sur ce soft qui ne détrône peut-être pas le hit de chez hit où de familiers barbares se décapitent (et dire


Tiens, prends ça dans la tronche !

que ce n'est qu'en 1999 qu'on imposa des lois contre les jeux à caractère "gore" !), mais qui a le mérite d'être très complet. Vous y reviendrez malgré tout.

Avant la dure épreuve des notes, je précise que vous trouverez un test et des cheat-modes pour ce même jeu dans le fanzine Another World 14 ainsi qu'une interview de Doc Barthoc (programmeur de FF2T) dans le numéro 15. Je cite ce dernier (AW15, p6, Q6) : "Les CPCistes sont des fainéants, ils ne terminent pas les jeux pour trouver tous les secrets (...)" alors... en tant que CPCiste, je suis bien placé pour confirmer cette affirmation !

GRAPHISME: à la Bollaware, clair et obscur à la fois... 17/20!

ZIK: simples, pas toujours les mêmes, appropriées, absentes pendant le jeu... 15/20!

BRUITAGES: on a vu mieux, mais il faut avouer que ce n'est pas facile... 12/20!

ANIMATION: très belle et finalement assez rapide quand on le sait !... 18/20!

JOUABILITE: à découvrir, donc pas géniale, mais c'est bien de chercher !... 16/20!

DIFFICULTE: on y revient, on progresse avec le temps, c'est dur... 17/20!

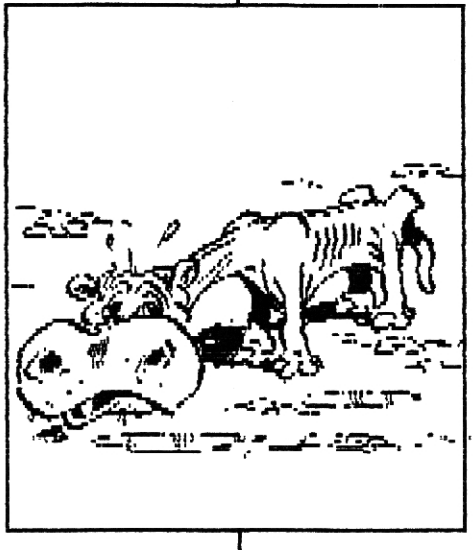
RICHESSE: plein de modes de jeu, y'a vraiment pas de quoi se plaindre, mais FF2T reste un jeu de baston... 18/20!

NOTE GLOBALE: (16/20!)

La compression sur Z80

Il existe deux problèmes intimement liés en informatique : la lenteur et le manque d'espace. C'est simple : si nous devons travailler avec beaucoup de données, on a besoin d'espace. S'il n'y a pas suffisament d'espace on travaille par petits paquets ; mais le traitement devient alors plus lent, car il est divisé en plusieurs petits traitements.

Heureusement, la lenteur a toujours été moins contraignante que le manque d'espace (notre cher Z80 exécute tout de même 4 millions de cycles par seconde, mais il travaille avec seulement 65536 adresses de mémoire ; en ce qui concerne les derniers membres de la famille i86... bon, nous ne sommes pas là pour parler des performances des différents microprocesseurs). Ça signifie qu'il est préférable de perdre un peu de temps pour optimiser l'espace. Le processus d'optimisation de l'espace est la Compression.



Il existe un grand nombre de méthodes de compression. Les trois méthodes les plus importantes sont la linéale, Huffman, et Lempel-Ziv.

Compression Linéale

La méthode la plus simple est la compression linéale. Un signal qui est répété plusieurs fois consécutives peut-être remplacé par deux signaux : le nombre de répétition et le signal lui-même. Pour permettre la décompression, un troisième signal d'identification doit être ajouté au début de la séquence.

Mais il y a un problème. Pour faire simple et optimiser l'enchainement des données et des signaux, le même répertoire de signaux doit être utilisé. Le problème apparait lorsque le signal représentant la compression existe aussi dans le répertoire original. La solution est peu agréable : chaque fois que le compresseur trouve dans les données le signal qu'il utilise pour la compression, il doit le traduire en un signal de compression, le nombre 1 (pour la répétition), et enfin le signal. Une optimisation est possible quand le signal de compression apparait deux fois : signal, nombre 2, signal.

Cette méthode est très simple, et l'implémentation est immédiate. En plus, si le signal de compression a été bien choisi, les résultats sont bons.

Comment choisir le signal de compression ? Cherchez le signal le moins répété (NDOffseT : de façon non consécutive) dans les données. C'est de cette manière que le problème de la répétition est réduit au minimum.

Compression d'huffman

La méthode linéale fait la compression des données au fur et à mesure qu'elle les lit, et le signal de compression est choisi "a priori". C'est bien, mais pas optimal. Une méthode examinant les statistiques des données serait meilleure, n'est-ce-pas ?

La compression d'Huffman fait comme ça : d'abord on lit les données et on compte le nombre de fois que chaque signal apparaît. Ensuite, le signal le plus abondant est remplacé par un nouveau signal de taille minimale, le deuxième plus abondant par un autre signal de taille un peu plus grande... jusqu'au signal le moins abondant, remplacé par le signal de taille maximale.

Cette méthode donne toujours le résultat minimum pour une équivalence signal à signal, mais la construction de la liste des nouveaux signaux est assez compliquée, car lle doit être faite d'après les résultats statistiques. En résumé, les signaux avec le double de fréquence deviennent des signaux de longueur deux fois moindre, et vice-versa. Un autre problème vient du fait que les ordinateurs ne travaillent pas avec de longues chaînes de bits, mais avec de petits paquets, toujours de la même longueur (8 bit dans le cas du Z80), et alors la compression et la décompression deviennent lourdes.

En plus, l'efficacité dépend de la taille du répertoire des signaux : soit n le nombre de signaux du répertoire, le résultat minimum est tout simplement log2(n) fois plus petit que les données originales. Ça signifie qu'avec n=16 les données ne sont plus compressées qu'au 1/4 ; avec n=256, au 1/8, etc...

Compression de Lempel-Ziv

Revenons à l'idée des répétitions consécutives. Elle est bonne, mais avec un texte comme celui-ci, elle n'arriverait pas à compresser quoique ce soit, car il n'y a pas de groupes de trois fois la même lettre... Mais si on examine le texte, il y a beaucoup de mots qui se répètent. Est-il possible de profiter de cette répétition non linéale ?

La réponse nous est données pas la compression de Lempel-Ziv. C'est simple : si la répétition d'une chaîne de signaux est trouvée, la répétition est remplacée par une référence à la première chaine du même type.

L'implémentation peut-être faite de deux façons. La première construit un "dictionnaire" de chaines, chaque chaine étant symbolisée par un signal ; ainsi, la compression et la décompression deviennent des traducteurs via le dictionnaire. Cela semble simple, mais la construction du dictionnaire est difficile car il doit être complet (pas de signaux sans traduction) et optimal (les chaînes les plus longues possibles).

La deuxième méthode prend pour base la compression linéale, le signal de compression. On examine les données et, chaque fois qu'une chaîne apparait répétée, elle est remplacée par : un signal de compression, l'index, et la longueur de la chaîne. Naturellement, le signal de compression est choisi avec attention. Cete méthode a un autre avantage : la compression linéale est automatisée, et d'une façon très intéressante. Voici un exemple :

AAAAAAAAAAAA -> A + SIGNAL + -1 (index relatif) + 11

ABABABABABAB -> AB + SIGNAL + -2 (index) + 10 (longueur)

ABCABCABCABC -> ABC + SIGNAL + -3 (index) + 9 (longueur)

Le problème de l'apparition du signal de compression peut-être résolu avec une valeur nulle pour l'index et la longueur.

FinaLement

Voici un petit programme pour votre CPC. Il utilise la compression linéale avec des données en mémoire et un fichier ascii. La syntaxe est la suivante :

CALL &BE80,adresse,longueur,@resultat%

Cette routine compresse linéairement les données placées à "adresse" jusqu'à "adresse+longueur-1" dans un fichier crée avec OPENOUT"fichier" ; la longueur compressée est restituée dans la variable "resultat%", une variable de type entier (NDOffseT : la variable doit exister lorsqu'on la passe en paramètre, faites par exemple un resultat%=0). Le fichier résultant peut-être fermé avec CLOSEOUT, ou peut rester ouvert pour recevoir plus de données compressées.

CALL &BE80,adresse,longueur,0

Dans ce cas, on compresse le contenu d'un fichier ouvert avec OPENIN"fichier", à "adresse" jusqu'à "adresse+longueur-1". Le fichier peut être fermé avec CLOSEIN ou rester ouvert pour continuer le travail.

Le signal de compression utilisé est &C7 car c'est le moins utilisé des codes du Z80 (c'est RST 0 !).

CNGSOFT

```
; Programme d'exemple de compression de données
; Par CNGSoft pour Quasar CPC 18

        Org &be80
cmpbyte equ &c7
putbyte equ &bc95
getbyte equ &bc80

        cp 3
        ret nz
        ld c,(ix+0)
        ld b,(ix+1)
        ld e,(ix+2)
        ld d,(ix+3)
        ld l,(ix+4)
        ld h,(ix+5)
        ld a,c
        or b
        jr z,g1
        push bc
        call p1
        pop hl
        ld (hl),c
        inc hl
        ld (hl),b
        ret
g1      jmp g2
p1      xor a
        ld c,a
        ld b,a
        ld (ix+0),a
p2      ld a,d
        or e
        jr z,p4
        ld a,(hl)
        inc hl
        dec de
        cp (ix+1)
        jr nz,p3
        inc (ix+0)
        ld a,(ix+0)
        xor &ff
        call z,p5
        jr p2
p3      call p5
        dec hl
        ld a,(hl)
        ld (ix+1),a
        inc hl
        ld (ix+0),1
        jr p2
p4      call p5
p5      ld a,(ix+1)
        cp cmpbyte
        jr z,p6
        ld a,(ix+0)
        and a
        jr z,p6
        cp 3
        jr c,p7
p6      ld a,(ix+0)
        and a
        jr z,p8
        ld a,cmpbyte
        call putbyte
        inc bc
        ld a,(ix+0)
        call putbyte
        inc bc
        ld a,(ix+1)
        call putbyte
        inc bc
        xor a
        ld (ix+0),a
        jr p8
p7      ld a,(ix+1)
        call putbyte
        inc bc
        dec (ix+0)
        jr nz,p7
p8      ret
g2      ld a,d
        or e
        jr z,g5
        call getbyte
        cp cmpbyte
        jr z,g3
        ld (hl),a
        inc hl
        dec de
        jr g2
g3      call getbyte
        and a
        jr z,g5
        ld b,a
        call getbyte
g4      ld (hl),a
        inc hl
        dec de
        djnz g4
        jr g2
g5      ret
```

Assembleur

HARDWARE

Assembleur

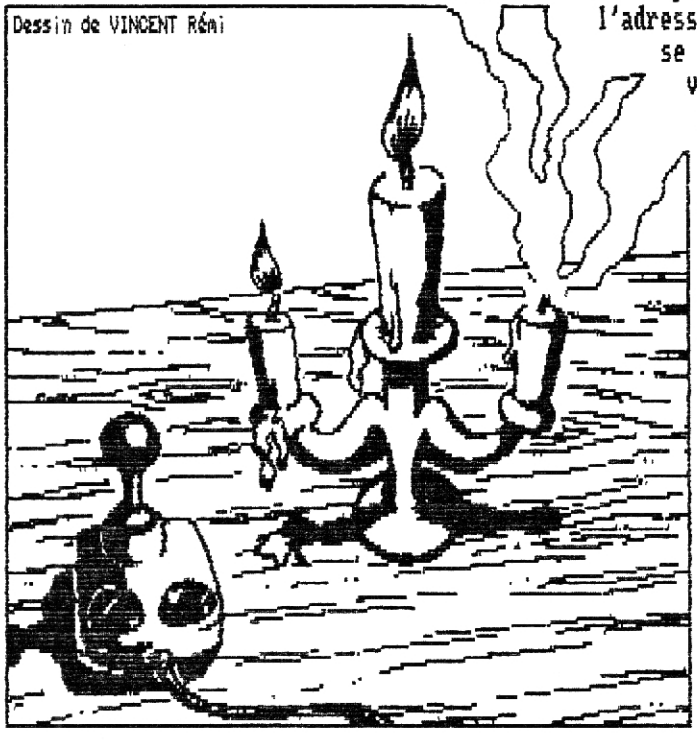
Une fois de plus, je me suis creusé la tête pour vous trouver un sujet intéressant pour cette nouvelle édition de la rubrique Hardware. La dernière fois, je vous avais parlé des boucles et de leur optimisation sur Z80 ; eh bien cette fois-ci nous allons explorer ses interruptions. Oui, le petit blond à lunettes a raison, j'ai déjà parlé de ça dans un très vieux numéro de Quasar CPC... mais il s'agissait d'un article spécifique au mode d'interruption vectorisé (IM 2) et depuis lors, j'ai de nouvelles informations à vous transmettre. De plus, une carte fort intéressante, la CPC-ISA de Siou, utilise largement les interruptions pour son fonctionnement.

Aussi, un fois n'est pas coutume, cette rubrique sera en fait plus une notice d'information qu'un cours.

C'est quoi une interruption ?

Pour que les choses soient claires, nous allons commencer par revoir le principe de base. Typiquement une interruption est un signal externe au microprocesseur qui permet d'en modifier le fonctionnement. On entend par là que lorsqu'une interruption est générée par un périphérique et est acceptée par le microprocesseur, celui-ci va exécuter un autre code avant d'acquitter l'interruption (indiquer au périphérique que l'interruption a été gérée) et de retourner à son code d'origine. La manoeuvre d'acquittement est spécifique au périphérique : sur un CPC standard aucun acquittement d'interruption n'est nécessaire, sur CPC+ celui-ci peut se faire via le registre Asic DCSR (suivant la configuration du bit 0 du registre Asic IVR), sur la CPC-ISA ça se négocie directement avec la carte ISA concernée, etc...

La plupart des microprocesseurs acceptent différents types d'interruptions et le Z80 n'a pas failli à cette règle puisqu'il en accepte deux : les interruptions non masquables et les interruptions masquables.



Dessin de VINCENT Rémi

Les interruptions non masquables

Il s'agit d'interruptions qui sont obligatoirement acceptée par le processeur - celui-ci ne peut pas les ignorer. Lorsqu'une requête d'interruption non masquable est envoyée au Z80, celui-ci termine l'instruction en cours puis effectue un RST &66. A l'adresse &66 doit donc se situer la routine (ou vecteur d'interruption non masquable) qui devra gérer l'événement. Une fois le traitement achevé, la routine devra impérativement faire un retour au programme principal via l'instruction RETN et non RET ou RETI ! Cette instruction, en plus de recharger dans PC la dernière valeur empilée, va reconfigurer le Z80 pour qu'il puisse de nouveau accepter des interruptions non masquables. En effet, une fois que le Z80 a accepté une telle interruption il refusera toutes les requêtes suivantes jusqu'à ce l'instruction RETN soit exécutée.

Sur CPC, ce type d'interruption n'est pas utilisé par le hardware interne, mais est laissé libre sur le port expansion et est géré par la Multiface II. Lorsque vous activez celle-ci, elle fait une requête d'interruption non masquable, attend que le Z80 finisse son instruction en cours et commute sa page ROM entre &0000 et &1FFF et sa page RAM entre &2000 et &3FFF. De ce fait, lorsque le Z80 fait effectivement son RST &66 en réponse à la requête d'interruption, il saute dans la ROM de la Multiface II... la suite, vous la connaissez.

Dans la pratique, ce type d'interruption est parfaitement adapté à des systèmes temps-réel.

Les interruptions masquables

On retrouve ici les interruptions standard du CPC. Comme le fait remarquer fort judicieusement le petit blond à lunettes, ces interruptions sont dites masquables car elles peuvent être refusées par le Z80 dans le cas où celui-ci est sous DI (Disable Interrupt). En outre, contrairement aux interruptions non masquables, à partir du moment où le Z80 est sous EI (Enable Interrupt) il acceptera toutes les interruptions qui se présentent même si on est en cours d'exécution d'une routine d'interruption. Ceci n'exclu toutefois pas la nécessité d'un acquittement d'interruption pour certain périphériques spécifiques (Asic des CPC+, CPC-ISA, etc...).

Toutefois, vous devez faire attention au fait que lorsque vous venez d'entrer dans une routine d'interruption, le processeur passe automatiquement en DI. Il ne faudra donc pas oublier de le remettre en EI avant de retourner au programme principal, sinon le Z80 n'acceptera plus aucune interruption ! En revanche, un RET suffit en fin de routine d'interruption ; pas d'instruction spécifique comme le RETN vu précédemment...

Le petit blond à lunettes est en train de s'agiter : et le RETI alors, c'est kwa ? Comme son nom l'indique, le RETI est l'instruction de RETour d'Interruption mais elle est inutile au regard du hardware du CPC, du CPC+, ou même de la CPC-ISA car elle est dédiée au périphériques Zilog (Z80CTC, Z80DMA, Z80SIO, etc...). Dans ce cas, son rôle est d'acquitter l'interruption ; mais, je le répète, ceci ne fonctionne que pour les périphériques Zilog, dans tous les autres cas le RETI se comporte comme un RET classique (sauf qu'il est plus lent).

Il est par ailleurs important de noter que le Z80 dispose de 3 modes de fonctionnement des interruptions bien distincts choisis à l'aide de l'instruction IM (Interrupt Mode).

Le mode d'interruption 0

Le mode d'interruption 0 (IM 0), qui n'est là que pour la compatibilité avec le processeur 8080 d'Intel, est complètement obsolète et ne mérite pas qu'on s'y intéresse. Dans ce mode de fonctionnement, lorsque le Z80 a accepté une interruption, c'est au périphérique de lui donner les instructions à exécuter. Il n'existe pas à ma connaissance de périphérique utilisant cette "fonctionnalité" sur CPC.

Le mode d'interruption 1

Il s'agit là du mode standard utilisé par le firmware. Lorsque le Z80 entre en interruption, il effectue un RST &38. Sur CPC, le seul périphérique générant des interruptions en interne est le Gate Array. Trois cents fois par secondes notre petit Z80 saute ainsi en &38 où le firmware a placé son vecteur d'interruption pour gérer le son, les couleurs, le lecteur de disquettes, etc... Le principal inconvénient de ce mode de fonctionnement vient principalement du fait que tous les périphériques génèrent une interruption à la même adresse (&38)... Le petit blond à lunettes a une fois de plus tenté de faire une remarque constructive ; oui, comme sur CPC seul le Gate Array fait des interruptions, ce mode pourrait suffire, mais ne serait-il pas tout de même intéressant de pouvoir choisir l'adresse de notre saut d'interruption ? Et puis, sur CPC+, on a 4 générateurs d'interruptions (les 3 DMA et le Raster (ou Gate Array)). De plus, lorsqu'on veut connecter une CPC-ISA (ou toute autre carte générant des interruptions), il est extrêmement intéressant de pouvoir différencier les interruptions au niveau hardware ; dans le cas contraire, tout le monde se retrouve en &38 et c'est au programme de tester qui a fait l'interruption... pas très commode n'est-ce-pas ?

La technique utilisée par le firmware pour différencier les interruptions internes des interruptions externe est assez primitive... Il a été décidé que les périphériques externes doivent maintenir la requête d'interruption (signal INT du Z80) jusqu'à que l'acquittement ait été effectué alors que le Gate Array ne génère qu'une impulsion. Ainsi, une fois qu'une interruption a été acceptée, la routine d'interruption du système se replace sous EI. Si c'est le Gate Array qui avait généré l'interruption, il n'y a plus de requête d'interruption en cours et rien ne va se passer... si c'est un périphérique externe qui avait provoqué le RST &38 (par exemple une carte ISA), un nouveau saut en &38 va aussitot avoir lieu ! Le petit blond à lunette commence à s'affoler : on empile les interruptions, ça va planter !!! Mais non, pas de panique, car avant de se placer sous EI dans sa routine d'interruption, le système jongle avec la carry secondaire afin de savoir s'il s'agit d'une nouvelle interruption ou non et, le cas échéant, saute en &3B, le vecteur d'interruption réservé aux périphériques externes.

Hum ! Tout ceci est bien compliqué et nous fait perdre pas mal de temps CPU alors qu'il serait tellement plus simple de différencier nos périphériques matériellement. De plus, cette méthode n'est pas exploitable avec les DMA du CPC+ puisque ceux-ci ne génèrent pas d'interruption "longue" mais seulement une impulsion tout comme le Gate Array.

Le mode d'interruption vectorisé

Voici le mode d'interruption (IM 2) le plus intéressant ! Il nous permet en effet de choisir l'adresse du vecteur d'interruption pour 128 périphériques différents via une table de vecteurs d'interruptions : le must. Pratiquement, une fois que le Z80 est en IM 2, lorsqu'une interruption se produit il forme une adresse à l'aide du registre I (octet de poids fort) et de l'octet placé sur le bus de données par le périphérique ayant demandé l'interruption. L'adresse ainsi obtenue lui donne l'endroit où aller lire l'adresse du vecteur d'interruption à charger dans PC. En clair, en se permettant quelques barbarismes mémoniques, notre Z80 fait :

```
PUSH PC
LD N,valeur sur le bus de données
LD PC,(IN)
```

Mais il y a un hic... le hardware de notre bon vieux CPC n'est pas du tout prévu pour fonctionner de la sorte. C'est-à-dire que lorsqu'il génère une interruption, notre Gate Array ne fourni aucun octet au Z80 pour déterminer l'adresse du vecteur d'interruption. Je vous passe les détails électroniques, mais dans une telle situation le Z80 devrait rester en haute impédance et capter une pseudo valeur &FF pour N. Ceci est effectivement le cas mais, pas toujours ! Lors de mon précédent article sur le mode d'interruption vectorisé je vous avais indiqué que ceci dépendait du type de CRTC ; après une étude plus poussée (sur types 0, 1 et 2), il s'avère que c'est bien plus compliqué que ça.

Pour résumer, disons que si vous utilisez votre CPC normalement (pas de bidouilles hard au niveau du CRTC), il saute tout le temps au dernier vecteur de votre table (N=&FF). Mais, pour une raison qui m'échappe, certains CPC sautent ailleurs de temps en temps. Sur mon bon vieux CRTC type 1 le mode IM 2 est stable en &FF mais sur celui que j'utilise à l'Armée ça se passe beaucoup plus mal avec des plantages sauvages sans raison apparente. En fait, je pense que le problème vient plus du couple Gate Array / CRTC que du CRTC lui-même. Mais je n'ai ni les compétences ni le matériel pour pouvoir approfondir la question.

La solution

Si vous désirez utiliser le mode vectorisé sur un CPC de base juste pour pouvoir déplacer le vecteur d'interruption, mon précédent article sur la question reste la solution idéale. Si vous avez un CPC custom, bonne nouvelle, lorsqu'on branche des cartes additionnelles aux CPCs "instables" il deviennent, semble-t-il, stables (ça a marché pour 2 CPC). L'utilisation de la CPC-ISA en mode vectorisé n'est donc pas compromise du tout sur CPC vieille génération !

Le CPC+ dans tout ça ?

N'en déplaise au petit blond à lunettes, le CPC+ est parfaitement étudié pour travailler un mode vectorisé grâce au registre IVR de l'Asic qui permet de choisir l'adresse de départ de la table des vecteurs d'interruption pour les DMA et le Raster. Mais je ne vais pas développer ce sujet ici car je l'ai déjà fait dans un précédent article de la rubrique CPC plus. Sachez tout de même que le mode vectorisé du CPC+, qui est géré normalement, est parfaitement incompatible avec celui que l'on peut improviser sur CPC vieille génération (la CPC-ISA est dessinée pour tolérer les deux modes de fonctionnement).

Programmons un petit peu

D'aucuns trouveront cet article un peu trop théorique. J'en suis désolé mais la place me manque pour y glisser quelques listings. Toutefois, toutes les informations nécessaires sont là et il ne tient qu'à vous de les mettre en pratique ! De plus, tous les listings nécessaires comme ceux qui installent le mode vectorisé pour CPC old, pour CPC+ ou qui programment un mode vectorisé compatible tous CPC sont déjà parus dans de précédents numéros de Quasar CPC.

Enfin, je vous conseille de remplir systématiquement toutes les adresses de la table des vecteurs d'interruption non utilisées par un vecteur de "guru". C'est à dire une routine qui vous avertira qu'un saut erroné a eu lieu et évitera tout plantage ; si tout se passe bien cette routine se sera jamais exécutée, mais l'expérience m'a prouvé que c'était parfois bien utile (surtout quand on tititille la CPC-ISA) ! Si vous utilisez Maxam je vous conseille une routine du genre :

```
Guru    BRK     ; Maxam Breakpoint routine
        EI      ; Notamment celle de Maxam 1.5
        RET     ; qui est très bien faite
```

Je vous encourage grandement à vous familiariser avec le mode d'interruption vectorisé car il va sans aucun doute être de plus en plus utile. Tout d'abord, il y a déjà la CPC-ISA qui n'est réellement exploitable qu'en mode vectorisé sous peine de perdre trop de temps à trier les interruptions de manière logicielle. Mais mon petit doigt me dit que d'autres cartes "vectorisées" pourraient ne plus trop tarder à sortir.

OffseT

HELPS

La Abadía del Crimen 2/3

Voici la suite des aventures de frère Guillaume d'Occan et de l'apprenti Adson de Melk... La difficulté du jeu va devenir terrible maintenant, une seule erreur, et l'abbé jetera Guillaume hors de l'abbaye, ou pire encore, Guillaume mourra !

Deuxième jour

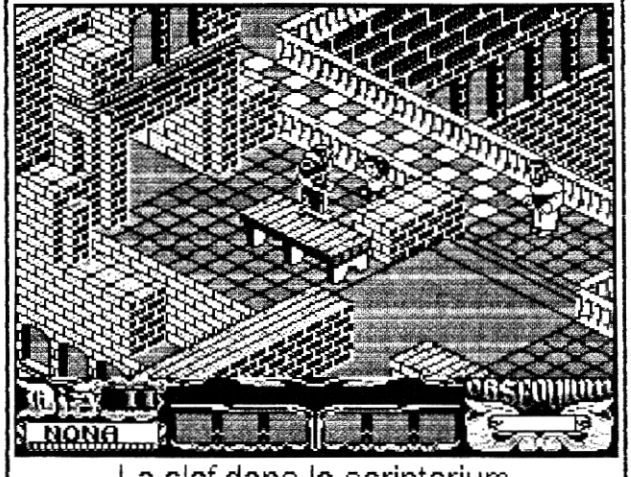
~prime

Mauvaise nouvelle : un inconnu a profité de la nuit et a volé les lunettes de Frère Guillaume ! Sans lunettes, Guillaume n'est pas capable de lire. Mais le temps manque pour démasquer de voleur :

Adson : "Debemos ir a la iglesia, maestro" - Nous devons nous rendre à l'église, maître.

Guillaume va à l'église. L'abbé a aussi des mauvaises nouvelles :

Abbé : "Hermanos, Venancio ha sido asesinado" - Frères, Venance a été assassiné.



La clef dans le scriptorium

~tierce

Abbé : "Venid aqui, fray Guillermo" - Venez ici, Frère Guillaume.

L'abbé veut lui expliquer quelque chose...

Abbé : "Debeis saber que la biblioteca es un lugar secreto. Solo Malaquias puedo entrar. Poseis iros" - Vous devez savoir que la bibliothèque est un lieu secret. Seul Malachias peut y entrer.

C'est maintenant le bon moment pour explorer la moitié Nord de l'abbaye. Sur le chemin, un moine demande l'attention de Guillaume :

Severin : "Venerable hermano, soy Severino, el encargado del hospital. Quiero advertiros que en esta abadia sucenden cosas muy extrañas. Alguien no quiere que los monjes decidan por si solos lo que deben saber" - Vénérable frère, je suis Severin, le responsable de l'hopital. Je veux vous avertir que dans cette abaye il y a des choses très étranges. Quelqu'un ne veut pas que les moines décident par eux-mêmes ce qu'ils doivent savoir.

Une étrange révélation n'est-ce-pas ? Guillaume traverse les portes que Malachias ferme chaque nuit, et il monte au scriptorium... Mais Malachias bloque une porte :

Malachias : "Lo siento, venerable hemano, no podeis subir a la biblioteca" - Désolé, vénérable frère, vous ne pouvez pas monter à la bibliothèque.

Mais il considère que Guillaume doit connaître le scriptorium :

Malachias : "Si lo deseais, Berengario os mostrara el scriptorium" - Si vous voulez, Berenger vous montrera le scriptorium.

Et Berenger, l'apprenti bibliothécaire, s'approche :

Berenger : "Aqui trabajan los mejores copistas de occidente" - Les meilleurs copistes d'occident travaillent ici.

Guillaume suit Berenger qui lui montre une table sur laquelle il y a un manuscrit et un livre :

Berenger : "Aqui trabajaba Venancio" - Venance travaillait ici.

~sexte

Adson : "Debemos ir al refectorio, maestro" - Nous devons aller au réfectoire, maître.

Guillaume suit Adson au réfectoire et se place devant le troisième plateau. Heure du repas !

Abbé : "Podeis comer, hermanos" - Vous pouvez manger, frères.

~neuvaine

Le repas est fini, et les moines se dispersent. Guillaume monte au scriptorium et essaie de prendre le manuscrit.

Berenger : "Dejad el manuscrito de Venancio o advertire al abad" - Laissez le manuscrit de Venance ou je vous dénoncerai à l'abbé.

Guillaume laisse le manuscrit. Mais c'est l'heure de commencer sérieusement l'enquête, et Guillaume doit s'emparer du manuscrit.

Pour commencer, Guillaume doit prendre la clé du bibliothécaire, placée sur la table près de la porte de la bibliothèque. Il ne peut pas la prendre directement car Malachias lui bloquerait alors le chemin. Il doit donc le distraire pendant qu'Adson prend la clé.

Ensuite, Guillaume se dirige vers la table de Venance. N'essayez surtout pas de prendre le livre sinon Guillaume mourra quelques secondes après : le livre a un terrible secret !

Pour s'emparer du manuscrit, Guillaume n'a plus qu'à attendre les cloches des Vêpres.

~vêpres

Adson : "Debemos ir a la iglesia, maestro" - Devons-nous nous rendre à l'église, maître ?

Berenger s'en va. Guillaume prend alors le manuscrit et se rend rapidement à l'église pour éviter que l'abbé se mette en colère.

Abbé : "Oremos" - Prions

~complies

La nuit tombe...

Abbé : "Podeis ir a vuestras celdas" - Vous pouvez vous retirer dans vos cellules.

~nocturne

C'est l'heure d'aller au dortoir... Dans le dortoir :

Adson : "Dormimos, maestro ?" - Devons-nous dormir, maître ?

L'enquête doit continuer : pressez "N" pour rester éveillé. Guillaume sort du dortoir, et un moine avec la tête couverte sort de la cellule de Malachias et Berenger. Guillaume le suit.

Un inconnu passe

Vous vous demandez comment Malachias peut revenir à l'église après avoir fermé la porte de la moitié Nord de l'intérieur... Voici la réponse : près de l'église, il y a une porte secrète ! La clé qu'Adson a prise dans la bibliothèque doit être utilisée maintenant. Faites attention ; c'est Adson qui porte la clé, non Guillaume. Le passage secret vous mène dans la cuisine.

Finalement, l'inconnu arrive au scriptorium, ramasse le livre, et s'enfuit. Pas de temps pour le suivre car le jour va arriver. Guillaume et Adson reviennent au dortoir. Adson demande une fois de plus à Guillaume s'il est possible de dormir : Oui, appuyez sur "S".

Troisième jour

~prime

Une fois de plus, le jour commence avec la messe de Prime. Pour résumer, les activités quotidiennes (messe de Prime, repas de Sexte, messe des Vêpres, sommeil de Nocturne) seront assurées.

L'abbé a des nouvelles inquiétantes :

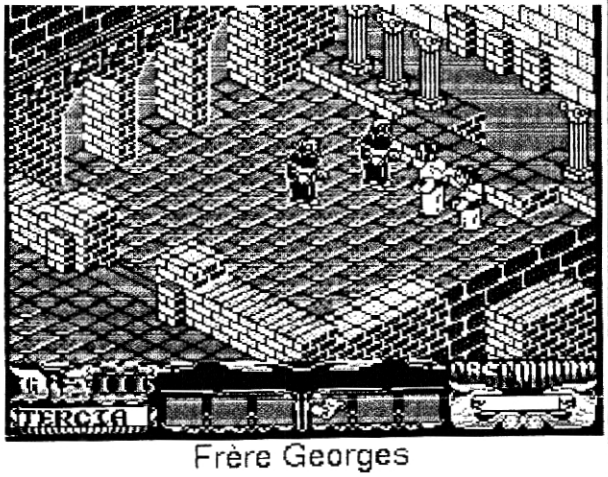
Abbé : "Hermanos, Berengario ha desaparecido. Temo que se haya cometido otre crimem" - Frères, Berenger a disparu. J'ai peur qu'un autre crime ait été commis.

~tierce

L'abbé a aussi des nouvelles pour Guillaume :

Abbé : "Quiro que conozcais al mas viejo y sabio de la abadia" - Je veux vous présenter l'homme le plus vieux et le plus savant de l'abbaye.

Et il conduit Guillaume jusqu'au couloir des dortoirs. Un nouveau moine est ici.



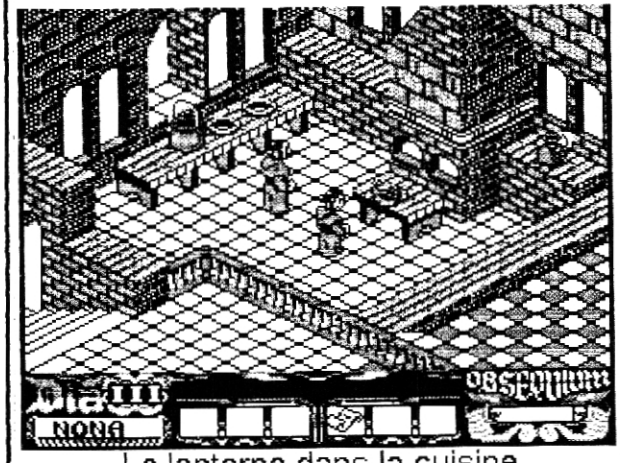
Frère Georges

Abbé : "Venerable Jorge, el que esta ante vos es fray Guillermo, nuestro huesped" - Vénérable Georges, je vous présente frère Guillaume, notre hôte.

Georges : "Sed bienvenido, venerable hermano ; y escuchad lo que os digo. Las vias anticristo son lentas y tortuosas. Llega cuando menos lo esparas. No desperdicieis los ultimos dias" - Soyez le bienvenu, vénérable frère ; et écoutez ce que j'ai à vous dire. Les voies de l'antéchrist sont impénétrables. Les choses arrivent quand vous vous y attendez le moins. Ne gâchez pas les derniers jours.

~neubaine

Après le repas de Sexte, quelqu'un a oublié une lampe à huile dans la cuisine. Adson doit la ramasser : celle-ci sera nécessaire plus tard.



La lanterne dans la cuisine

Quatrième jour

~prime

Le jour commence avec Abbey Morning News :)

Abbé : "Hermanos, han encontrado a berengario asesinado" - Frères, Berenger a été trouvé assassiné.

~tierce

Abbey Flash News maintenant :)

Abbé : "Ha llegado Bernardo, debeis abandonar la investigacion" - Bernard est arrivé, vous devez abandonner l'enquête.

Guillaume se rend au réfectoire mais Severin l'interpèle dans le couloir :

Severin : "Esperad, hermano" - Attendez, frère.

Severin vient de faire l'autopsie du corps de Berenger :

Severin : "Es muy extraño, hermano Guillermo. Berengario tenia manchas en la lengua y en los dedos" - C'est très étrange, frère Guillaume. Berenger avait des taches noires sur la langue et les doigts.

Des taches noires ? Elles viennent d'un livre : le livre qu'il avait volé deux nuits auparavant dans le scriptorium.

Pendant la conversation, Bernard Gui arrive à l'abbaye.

~neubaine

Bernard Gui veut remplacer Guillaume dans l'enquête :

Bernard Gui : "Dadme el manuscrito, fray Guillermo" - Donnez-moi le manuscrit, frère Guillaume.

Guillaume doit obéir et Bernard Gui lui confisque le manuscrit. Il s'en va ensuite chez l'abbé pour le lui donner.

~nocturne

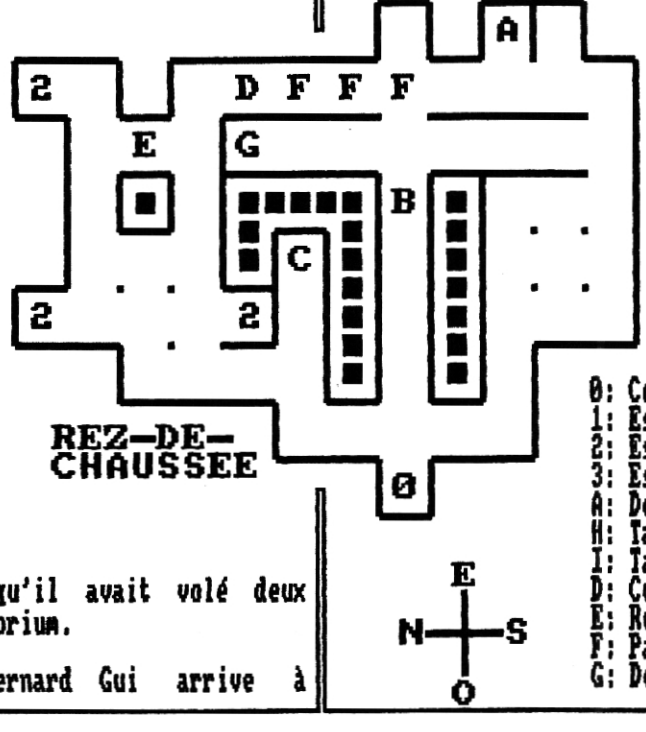
Pas de sommeil : l'abbé a oublié la clé de sa chambre dans l'église... et Guillaume doit profiter de l'occasion.

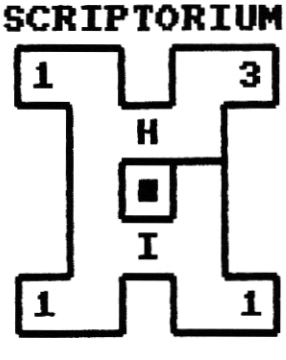
Pour arriver jusqu'à l'église sans être surpris par l'abbé, Guillaume doit sortir du dortoir et emprunter le second couloir. Il prend alors la clé et s'en retourne au dortoir par le premier couloir.

Bonne nuit !

A conclure...

CNGSOFT





0: Commencement du jeu
1: Escalier vers le rez-de-chaussée
2: Escalier vers le scriptorium
3: Escalier vers la bibliothèque
A: Dortoir de Guillaume et Adson
H: Table de Malachias
I: Table de Venance et Berenger
D: Cuisine
E: Réfectoire
F: Passage secret
G: Dortoir de l'abbé

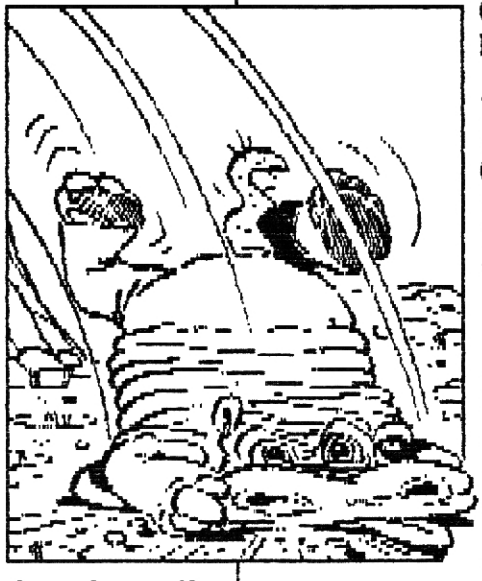
CODING

Assembleur

Cette rubrique Coding est encore dédiée au réseau Virtual Net. J'ai en effet un autre type de protocole à vous présenter. Celui-ci n'a pas été testé faute de temps et de matériel mais sa simulation sous Basic fonctionne à parfaitement !

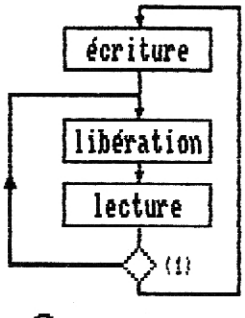
Du nouveau !

Pourquoi donc chercher un autre genre de protocole alors que celui déjà présenté ravit déjà un grand nombre de CPCistes lors les meetings ? Tout simplement parce que celui-ci à au moins un défaut : il monopolise totalement le Z80 lors des longues périodes de synchronisation. Ceci empêche notamment l'usage de techniques sympathique comme les rasters, les changement de mode et gène l'émission de sons. Bref, on est coincé dès lors que l'on veut exécuter une routine à une fréquence régulière.



L'idée est donc de garder les interruptions en permanence, la communication se fera entièrement sous interruption. Un inconvénient de cette façon de procéder est l'augmentation non négligeable du temps de communication étant donnée la fréquence faible des interruptions du CPC (300 Hz). De plus ce temps grandit avec le nombre de CPC qui échangent des données, c'est-à-dire avec le nombre de joueurs.

Voici la première méthode sur ce principe qui vient à l'esprit :



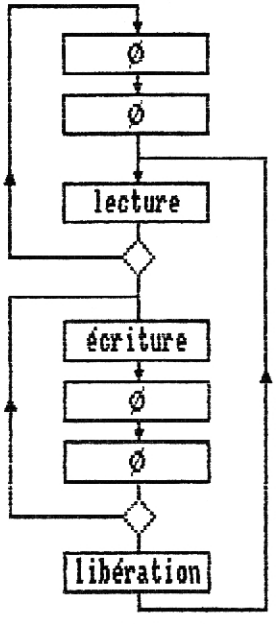
- écriture = envoi successif des n bits du message (un à chaque interruption)
- libération = mise à l'état neutre de la ligne (écriture d'un 1)
- lecture = réception des n bits du message (un à chaque int.)

Comme dit plus haut on ne réalise qu'une seule opération à chaque interruption. Comme d'habitude ce protocole suppose que l'on connait le nombre de joueurs et que chaque ordinateur sait quand c'est à lui de parler (ce qui correspond à l'étape d'écriture). Il faut évidemment que chaque CPC démarre dans ce graphe par l'opération adéquate, déterminée par son numéro de joueur (dans ce cas soit la lecture, soit l'écriture).

Ceci dit, la séquence telle quelle ne marche pas ! De temps en temps on observe une erreur de transmission. Elle est due à une lecture prématurée de la ligne alors que l'écriture n'est pas terminée ou à une libération de la ligne alors que la lecture n'est pas finie.

Pour éviter tout problème, sans trop compliquer le protocole, il vient l'idée d'ajouter des étapes intermédiaires d'inactivité pour assurer que la lecture ait toujours lieu quand la ligne est dans un état stable. Voilà ce que ça donne :

il y a en fait à ce niveau 2 boucles imbriquées, l'une pour le nombre de bits et l'autre pour le nombre de CPC.



une seule boucle ici, pour le nombre de bits à envoyer (le même qu'en lecture bien entendu !).

Attention, la "légende" est différente que précédemment :

- écriture = envoi d'un bit du message (mise à 1 ou Ø de la ligne)
- libération = mise à l'état neutre de la ligne (écriture d'un 1)
- lecture = réception d'un bit du message
- Ø = ne rien faire à cette interruption là !

Bien sûr, l'insertion d'états supplémentaires augmente la durée d'un cycle d'échange d'informations mais elle rend celui-ci fiable.

J'ai développé ci-dessous un cycle d'échange entre 3 ordinateurs d'informations codées sur 2 bits, utilisant le schéma précédent. Les étapes sont représentées au même niveau pour plus de clarté mais en réalité les interruptions n'ont pas lieu au même instant sur tous les CPC (ce serait trop facile).

CPC no1	CPC no2	CPC no3
ecr1	ø	lib
ø	lect1	lect1
ø	ø	ø
ecr2	ø	ø
ø	lect2	lect2
ø	ecr1	ø
lib	ø	ø
lect1	ø	lect1
ø	ecr2	ø
ø	ø	ø
lect2	ø	lect2
ø	lib	ecr1
ø	lect1	ø
lect1	ø	ø
ø	ø	ecr2
ø	lect2	ø
lect2	ø	ø

temps

Quel temps fait-il ?!

On voit qu'un échange de mots de 2 bits à 3 orditeurs donne un cycle qui dure 17 interruptions, ce qui réprésente à quelque chose près la durée de trois rafraichissements d'écran. Donc à priori avec ce principe c'est raté pour des jeux où l'affichage des actions de joueurs est à 50Hz.

Avec toujours 3 ordinateurs qui échangent 8 bits (ce qui est beaucoup, je pense que 4 bits suffisent pour un bon nombre de jeux) il faut compter 71 interruptions par cycle, soit environ un quart de seconde.

Je pars du principe qu'on utilise la seule interruption disponible sur CPC, à 300Hz, mais le schéma est censé fonctionner quelle que soit la fréquence.

D'ailleurs, vu la fréquence faible de variation de l'état de la ligne avec ce protocole, on peut espérer le faire fonctionner sur CPC+ en y apportant éventuellement quelques modifications.

En vrac

Une subtilité que je n'ai pas encore évoquée peut faire capoter l'ensemble. Les CPC n'ont pas exactement la même horloge (encore plus entre CPC et CPC+). Donc, il faut prendre soin de recaler les CPC de temps en temps sur l'étape adéquate. D'après des calculs fumeux, un recalage toutes les 5s environ suffit dans tous les cas de figure.

La communication étant sous interruption on peut créer une interface qui la fasse apparaitre au programme principal comme un UART...

Vous avez ci-contre un prog qui simule le protocole, il est mal commenté et un peu brouillon mais il peut vous intéresser.

Zik

```
10 DEFINT a-z
20 MODE 2
30 'FN1 est l'état de la ligne
40 'o(1), o(2) et o(3) sont les états des 3 cartes réseau
50 DEF FN1=o(1) AND o(2) AND o(3)
60 '
70 FOR n=1 TO 3:o(n)=1:NEXT ' ligne au repos
80 '
90 code=&D6 'code à échanger
100 n1=1:n2=1:n3=1
110 b1=0:b2=0:b3=0
120 lib1=0:lib2=1:lib3=1
130 wait1=0:wait2=0:wait3=0
140 'n? numéro de l'émetteur en cours (1 à 3)
150 'b? numéro du prochain bit à envoyer (0 à 7)
160 '
170 EVERY 12,3 GOSUB 210
180 FOR x=1 TO 100:NEXT:EVERY 12,2 GOSUB 440
190 FOR x=1 TO 100:NEXT:EVERY 12,1 GOSUB 650
200 GOTO 200
210 '** ordinateur 1 **
220 PRINT "n1=";n1
230 IF n1=1 THEN GOTO 330
250 IF lib1<>0 THEN lib1=lib1-1:o(1)=1:PRINT"p":RETURN
260 'reception                          ← 'liberation
270 PRINT"Lect"
280 i1=i1+FN1*2^b1
290 b1=(b1+1) AND 7
300 lib1=2
310 IF b1=0 THEN n1=(n1 MOD 3)+1:PRINT "reçu par 1 :";
    HEX$(i1):wait1=0:i1=0
320 RETURN
330 'ecriture
340 IF wait1<>0 THEN wait1=wait1-1:PRINT"p":RETURN
350 PRINT"Ecr"
360 o(1)=-((code AND 2^b1)<>0)
370 b1=(b1+1) AND 7
380 wait1=2
390 IF b1<>0 THEN RETURN
400 n1=(n1 MOD 3)+1
410 lib1=3 'il faudra faire la libération au prochain coup
420 RETURN
430 '
440 '** ordinateur 2 **
450 IF n2=2 THEN GOTO 550
460 ' lecture
470 IF lib2<>0 THEN lib2=lib2-1:o(2)=1:RETURN'liberation
480 'reception
490 PRINT"lect2"
500 i2=i2+FN1*2^b2
510 b2=(b2+1) AND 7
520 lib2=2
530 IF b2=0 THEN n2=(n2 MOD 3)+1:PRINT "reçu par 2 :";
    HEX$(i2):wait2=0:i2=0
540 RETURN
550 'ecriture
560 IF wait2<>0 THEN wait2=wait2-1:RETURN
570 o(2)=-((code AND 2^b2)<>0)
580 b2=(b2+1) AND 7
590 wait2=2
600 IF b2<>0 THEN RETURN
610 n2=(n2 MOD 3)+1
620 lib2=3 'libération au prochain coup
630 RETURN
640 '
650 '** ordinateur 3 **
... idem que ** ordinateur 2 ** en remplacant :
    n2    -> n3
    lib2  -> lib3
    o(2)  -> o(3)
    i2    -> i3
    b2    -> b3
    wait2 -> wait3
```

CPC plus

Greyscale

Alors même que tous les CPCistes veulent mettre de plus en plus de couleurs dans leurs réalisations, je vous propose, en rebelle que je suis, d'en mettre moins. Ou plutôt de convertir une bien belle palette de couleurs en infâmes niveaux de gris.

C'est un effet très joli, assez facile à réaliser et en plus, ça tombe bien, j'avais besoin d'un tel effet pour Stormlord+. Le principe est très simple. On prend la palette en cours, on fait deux/trois petits calculs et on balance le tout à l'Asic. Bon courage.

SNN

Hein ? Quoi ? J'ai pas fini mon boulot ? Alors voyons comment faire les calculs. Comme tout le monde (sauf le petit blond à monocle (oui, il avait une paire de lunettes normale, mais en prévoyant que je suis, j'ai préféré lui filer un coup de poing salvateur avant même le début de mon article)) le sait, une palette en niveaux de gris (d'où "GREYSCALE") se compose de valeurs de rouge/vert/bleu identiques. Par exemple, un gris moyen (50%) sur CPC+ sera &7, &7 et &7. Soit on balance à l'Asic &77 au premier octet et &07 au deuxième. Le traitment de la palette initiale sera vu dans le prochain numéro.

SNN

Je peux toujours pas partir ? Bon. Continuons alors. Comme vous pouviez vous en douter, il ne suffit pas de faire la moyenne des valeurs rouges, vertes et bleues pour obtenir la valeur à poker. Non, c'eut été trop simple. Et c'est pour cela que les ignares comme Tony ne peuvent pas saisir la substantifique moelle de cet article délicieusement rédigé. En fait, on se rend compte que si sur un écran on dessine trois carrés juxtaposés, l'un rempli de vert, un autre de rouge, le troisième de bleu, tous trois à la valeur 15, les intensités lumineuses de ces carrés ne sont pas identiques... Elles sont même décroissantes. Ainsi, il va falloir pondérer les valeurs de nos trois couleurs fondamentales, ce que nous avons vu page 46 du numéro 21 de ce merveilleux fanzine. A très bientôt.

SNN

C'est bon, lâchez ce flingue, je retourne devant mon CPC pour finir l'article... Pfiou... Je l'ai échappé belle ! Euh... où en étais-je ? Ah oui. Comme le vert est plus lumineux que le rouge, lui-même plus lumineux que le bleu, j'ai décidé de leur attribuer respectivement les coefficients 3, 2 et 1. Ainsi, je prends dans les registres de l'Asic le premier octet de l'encre qui m'intéresse. Mettons &6400 (fond de l'écran). Je sauve soigneusement l'octet et j'en prends les 4 derniers bits (4 à 7). Je RRAte le tout quatre fois et je double tout ça avec un ADD A,A bien senti. Résultat : La valeur de rouge est doublée, je suis heureux. J'y ajoute les quatre premiers bits de &6400 (la composante bleue) et je sauve. Pour le vert, même principe mais je multiplie par trois (voir le programme d'exemple) et finalement j'ajoute les jaunes d'oeufs à la farine et au sucre en poudre, lentement pour qu'il n'y ait pas de grumeaux. Oups, pardon, je m'égare.

Tiens, ça me fait penser que je n'ai pas rempli mon quota de vacheries anti-Tony dans cet article. Alors : "Tony, change de page, c'est des trucs pour grandes personnes, ici !".

Finalement, la somme n'est pas à diviser par 3, mais bien par 6. Or comme ce n'est pas très facile de trouver un moyen de diviser par 6, je vous propose de revenir en 6ème. Flash back (imaginez la maîtresse avec son morceau de craie, sa règle jaune de 1 mètre et son gros compas que vous rêviez de prendre pour lui planter dans l'oeil) "Diviser par 6 c'est soustraire 6, x fois." et x est le résultat. Fin du flash back.

Opérons donc.

Magie, un résultat compris entre 0 et 15. On trafique un peu pour que les quatre premiers bits soient recopiés sur les quatre derniers, on balance la purée en &6400 et comme &6401 tire la tronche, on fait de même pour lui. Résultat : ça marche et moi je vous souhaite une excellente soirée sur notre chaîne. Tout de suite, la météo.

SNN

C'est bon là, je peux partir ?
Merci, vous êtes bien urbain...

```
; Programme de conversion
;  de palette couleur en
;     niveaux de gris
;
; Par SNN pour Quasar CPC 18
;
Grey    xor a
        ld (compt),a
        ld hl,&6400
        ld (base),hl
boucle  ld hl,0
        ld de,(base)
        inc de

; Traitement du vert

        ld a,(de)
        and %1111
        ld c,a
        ld b,0
        add hl,bc
        add hl,bc
        add hl,bc
        dec de

; Traitement du rouge

        ld a,(de)
        and %11110000
        rra
        rra
        rra
        rra
        ld c,a
        add hl,bc
        add hl,bc

; Traitement du bleu

        ld a,(de)
        and %1111
        ld c,a
        add hl,bc

; Division par 6

        xor a
        ld bc,6
divise  sbc hl,bc
        inc a
        jr nc,divise
        ld (resultat),a
        or a
        jr z,annul
        dec a
        ld (resultat),a

; Recopiage bits 0-3 dans
; bits 4-7 puis envoi a
; l'Asic.

annul   ld b,a
        and %1111
        rla
        rla
        rla
        rla
        add a,b

        ld hl,(base)
        ld (hl),a
        inc hl
        ld (hl),a
        inc hl
        ld (base),hl

; On passe a l'encre
; suivante...

        ld a,(compt)
        inc a
        ld (compt),a
        cp 16
        jp nz,boucle
        ret

; Compteurs - Stockage

resultat nop
compt   nop
base    ds 2
```

QUASAR

DISC

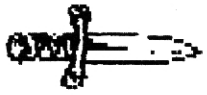
Tout comme la dernière fois nous vous proposons un Quasar Disc avec ce numéro 18 de Quasar CPC. En toute logique, étant donné que celui de la dernière fois portait le doux nom de Quasar Disc 1, celui-ci est donc le Quasar Disc 2 !

Qu'avons-nous à vous proposer cette fois-ci ? Hum ? Eh bien sur la face A vous trouverez les Aveux II, l'interview musclée de Candy par Tony, avec preuves à l'appui s'il vous plait ! Et sur la face B, outre tous les listings présents dans ces pages, vous aurez droit au kit de développement Small C pour CPC. Celui-ci est complet et entièrement configuré pour pouvoir être utilisé directement sous réserve que vous ayez deux lecteurs de disquettes : un pour le boot CP/M et l'autre pour le Small C.

De plus, veinards que vous êtes, ce Quasar Disc 2 contient la version alpha du Citron Boot, un utilitaire de boot configurable à souhait pour les heureux possesseurs de RAMCard !

Voilà, j'espère que vous pourrez tirer profit de ce nouveau Quasar Disc. Pour vous le procurer il vous suffit d'envoyer une disquette 3" ou 3"1/2 ainsi qu'un timbre à 4F50 (pour le retour) à la rédac'. Attention toutefois, le kit Small C étant assez gros, il n'est disponible que sur disquette 3"1/2. Si vous désirez le recevoir tout de même sur disquette 3" il vous faut alors en prévoir deux sinon nous ne pourrons pas vous copier la totalité du kit (il manquera les documentations).

OffseT



DOSSIER

Alors que de plus en plus de monde souhaite voir des jeux sur CPC, de moins en moins de jeux sortent. Situation assez paradoxale qui peut ma foi s'expliquer. En effet, après avoir demandé à plusieurs codeurs pourquoi ils ne se tournaient pas plus vers les jeux, la même réponse ressortait systématiquement : c'est trop long et trop compliqué. C'est pourquoi vous allez me retrouver (Hicks, c'est moi !) dans Quasar CPC durant quelques numéros, où je vais aborder les différents points qui vous permetteront de faire vos propres jeux.

Comme il faut bien commencer par quelque chose, voyons brièvement quelques points d'aspect généraux tout de même importants.

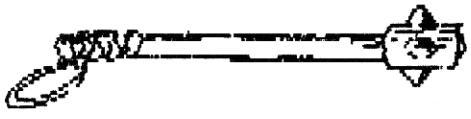
Le scénario

Trop souvent négligé, il m'est plutôt d'avis que le scénario fait partie des éléments essentiels. Donc, par pitiée, je ne veux plus voir de fichier ascii expliquant une histoire sans intéret (non, je ne vise personne ;). Prenez plutot exemple sur les productions allemandes (Megablasters, Black Land...) où les scénarios tiennent toujours debout et sont très bien mis en scène. Car actuellement, sur les machines soit disant "plus puissantes" combien de fois a-t-on pu voir des jeux techniquement corrects mais manquant cruellement d'intéret ? C'est d'autant plus vrai avec les jeux en 3D, soit dit en passant. Donc, pensez à faire un scénario original, évolutif et non linéaire. Je vous conseille par exemple de commencer avec un scénario anodin et de dévoiler la trame de l'histoire peu à peu sans pour autant embrouiller le joueur. Pensez également aux rebondissements qui sont essentiels pour la durée de vie... Maintenant, à vous de faire quelque chose d'intéressant, et n'oubliez pas que la puissance de la machine n'a ici aucune importance, c'est pourquoi vous êtes parfaitement en mesure d'égaler les scénarios des jeux dernier cri.

Retenir le joueur

Si vous ne voyez pas d'autres moyens pour retenir le joueur que de l'attacher à sa chaise avec des menottes, lisez ce qui suit. Eh oui, car pour intéresser le jeu, un bon scénario n'est pas tout. Une fois le jeu fini, le joueur risque de s'ennuyer ferme en recommençant une énième partie. Ça y est, à peine arrivé j'ai déjà cette godiche de petit blond à lunettes sur le dos pour raconter une fois de plus n'importe quoi en disant qu'il suffit de faire le jeu très dur pour retenir le joueur. Tiens, tiens, j'ai une piste : le petit blond à lunettes aurait des racines espagnoles. Quoi de plus énérvant qu'un jeu où on avance pas ? Le petit blond à lunettes ? Ok, je vous l'accorde... Mais trève de galéjades, pour retenir le joueur, il va falloir nous intéresser aux options.

Tout d'abord, pour que personne ne reproche à votre jeu d'être trop simple, il est impératif d'intégrer des niveaux de difficulté. De cette façon, tous les types de joueurs y trouveront leur compte.



Dans un autre domaine, prévoyez également un mode customisation (pour les jeux de voitures, combat, sport...) avec possibilité de sauvgarder la configuration. Par exemple, construire ses circuits, créer son personnage, son équipe, bref une option qui permettera à votre jeu de se renouveler sans cesse.

Et puis, le must, je veux bien entendu parler de mode multijoueur. Une telle option peut conférer un intéret illimité à tout type de jeu, c'est pourquoi il FAUT tout faire pour tenter d'intégrer cette possibilité... De plus, si votre jeu nécéssite des écrans indépendants, n'oubliez pas le réseau ! Je vous encourage d'ailleurs à suivre le dossier réseau que Zik a débuté lors du numéro 16. Malgré tout, ce n'est pas exploitable pour tous les types de jeux et c'est bien dommage.

L'algo général

Après avoir abordé ces quelques points d'aspect "philosophiques" (n'est-ce pas Offset ;), intéressons-nous à la programation de notre futur jeu. Prenons le cas le plus général, comprenant presque tous les types de jeux, nous aurons alors un algorithme de la forme suivante :

Frame
Sauvegarde décor
Restitution décor
Affichage sprite
Jouer musique/bruitages
Gestions diverses
(touches, collisions, compteurs...)

Ceci résume en gros ce qu'il va nous falloir gérer à chaque VBL, et dans cet ordre. En effet, tout ce qui concerne l'affichage doit être fait en priorité pour éviter de se manger le balayage. Imaginez que vous êtes en train d'afficher un sprite, et que le canon à électron passe par là avant même que vous ayez fini. Le résultat à l'écran sera un sprite affiché pas en entier mais à une vitesse de 50 images par secondes, cela sera traduit par un clignotement ou une sacade. Pour ce qui est de l'ordre Sauvegarde-Restitution-Affichage, je pense que vous avez tous compris qu'il vaut mieux avoir le moins de temps possible entre l'effaçage du sprite (par la restitution du décor) et l'affichage du sprite. La musique, elle, peut être jouée n'importe quand du moment que c'est à chaque VBL. La meilleure solution pour éviter à notre musique de ralentir consiste à la mettre sous interruption. Nous verrons ça une autre fois. Les diverses gestions peuvent également être misent sous interruption, mais à condition que votre jeu tourne au 50Hz.

La sauvegarde

Avant d'afficher notre sprite, il faut sauvegarder son futur emplacement afin de pouvoir le restituer par la suite. La première méthode qui est de loin la plus simple, consiste à sauvegarder la matrice rectangulaire correspondant à la taille de notre sprite, et cela aux nouvelles coordonnées. Il me semble que le LDI est l'instruction la plus adaptée à ce genre de transferts (voir listing 1). Le seul inconvénient vient du fait qu'il va nous falloir une routine différente pour chaque sprite de taille différente. C'est pas très grave car si on veut éviter le LDIR, c'est la seule solution...

Oui mais voila, j'entends déjà les perfectionnistes critiquer. Ils ont raison. Avec la méthode expliquée plus haut, nous allons sauvegarder du décor qui était déjà en mémoire. Un exemple : prenez un sprite de 8 octets de large se déplacant vers la gauche (octet par octet) et ayant le décor qu'il écrase déjà stocké en mémoire. Au prochain déplacement, seul le contenu des octets de la colonne à gauche de notre sprite diffèreront de ceux déjà stockés en mémoire. C'est pourquoi le mieux serait de ne sauvegarder que ceux-ci. Ca nous ferait gagner le temps du transfert de 7 octets fois la hauteur dans notre exemple ce qui est tout de même pas mal. Mais, dans le buffer, on ne peut pas mettre la nouvelle colonne de décor saugegardée toujours à la suite du décor déja stocké en mémoire car notre buffer aurait alors une taille indéterminée. Il va donc falloir nous servir de tests pour savoir où on est dans le buffer lors de la sauvegarde et de la restituion. C'est relativement exploitable, pas très long et peu contraignant niveau mémoire.

Mais je gardais la meilleure solution pour la fin. En fait, la technique est de carrément éliminer l'étape de sauvegarde. En stockant une deuxième fois votre décor en mémoire, un simple pointeur suivant les déplacements votre sprite indiquera à la routine l'adresse où est le décor à restituer. Ça prend évidement beaucoup de place, mais le résultat en vaut la peine, surtout si vous animez beaucoup de sprites. De plus, selon les jeux, vous n'êtes pas obligés de mettre tout le décor une deuxième fois en mémoire, mais seulement là où les sprites peuvent passer. Cette technique n'est malheureusement utilisable que dans les jeux sans scrolling constant (pour le style de Prehistorik 2 ça pourrait éventuellement fonctionner, ou dans un jeu écran par écran). En effet, pour les autres types de jeux ça prendrait sûrement plus de temps de sauvegarder la nouvelle colonne de décor apparue à l'écran grâce au scrol que de passer par la méthode classique.

La restitution

La technique précédemment évoquée étant impossible pour l'étape de restitution, il va falloir recourir aux LDIs. La routine sera d'ailleurs presque similaire au listing 1 à la différence que le transfert s'opère de la mémoire à la mémoire écran, ce qui va occasionner quelques modifications.

Nous allons maintenant voir ensemble 3 routines d'affichage différentes. Expliquons-les une par une.

L'autogénéré

~ Listing 2

Oui, ça prend beaucoup de mémoire. On peut dire qu'en gros, pour une routine autogénérée masquant à l'octet, elle va prendre trois fois plus de mémoire que le sprite que l'on veut afficher (sans tenir compte des tests d'octets vides, voir source). L'autogénéré, comme son nom l'indique, consiste à construire une routine d'affichage en incrustant les octets du sprite dans la routine. Cela nous évitera d'avoir un registre pointant sur le sprite, et du même coup, ça suprimera toutes les boucles.

Remarquez que j'ai choisi délibérément de faire le test de la hauteur avant le BC26, ceci nous évitera de descendre d'une ligne inutilement après avoir affiché le dernier octet du sprite. Comme vous pourrez le voir, cette routine fait un masquage à l'octet, qui, je le rappelle n'est utilisable qu'en mode 0 car dans les autres modes graphiques on risquerait parfois de se retrouver avec plus de pixels éteinds que d'allumés dans un octet : attroce.

Notez également qu'il vaut mieux ne pas générer une routine pour chaque sprite mais seulement pour ceux se déplacant. On peut alors travailler avec deux types de routines.

Le masquage pixel

~ Listing 3

Tout d'abord, sachez que ce type de routine prend pas mal de temps machine, c'est pourquoi je ne la vous conseille seulement si vous travaillez en mode 1 ou 2 (!). Le résultat en mode 0 ne valant pas vraiment le coup au niveau résultat/temps machine par rapport au masquage à l'octet.

Le principe n'est pas très compliqué. On prend les octets du masque et on fait un ET (je parle ici du AND et non d'un quelconque film) avec l'octet de la mémoire écran afin que seuls les pixels différents de ceux du sprites (donc définis par le masque) soient retenus dans l'accumulateur. Puis grâce au OR, on affiche les données du sprite là où il n'y a pas le masque (donc pas le décor) et on affiche l'octet. Enfin, on incrémente les 3 pointeurs respectifs.

Le masquage pixel prend pas mal de mémoire du fait qu'en plus de votre sprite, vous devrez aussi stocker le masque qui prend autant de mémoire.

Bien que cette technique prenne du temps (16 nops pour un octet, les boucles pouvant être recopiées), je vous conseille de l'adapter à l'autogénéré et on peut ainsi obtenir un résultat de 10 nops pour un octet, mais la mémoire consommée est encore plus importante que pour un masquage à l'octet. A vous de voir.

Le bitplan

~ Listing 4

Cette fois-ci c'est d'un masquage au pixel grâce à un simple OR dont nous allons parler. La contrainte réside dans le fait que la moitié des encres devront être résérvées pour le décor et l'autre moitiée pour les sprites. Ceci rend cette technique quasiment inutilisable en mode 2 et très difficilement applicable au mode 1. Le mode 0 reste donc la meilleure alternative.

Pour ce qui est du principe, c'est très simple. On récupère chaque octet du sprite, on fait un OU exclusif avec le décor, et on envoie le résultat de cette opération à l'écran. Grâce au jeu d'encres, pas d'équivoque possible, le sprite est masqué au pixel.

La restitution incrustée

Pour finir, voyons une méthode permettant de sauvegarder et d'afficher un sprite avec la même routine. Si nous faisions ces deux étapes séparément, il faudrait incrémenter deux fois nos pointeurs, faire deux BC26... bref, perte de temps. Je ne vais donc pas vous fournir le source de cette routine, d'une part car j'occupe déjà 4 pages, ce qui est beaucoup, et d'autre part pour vous faire chercher un peu, vous ne comprendrez que mieux. Je vous donne juste le morceau de programme permettant d'afficher un octet :

```
ld a,(de)     ; HL=sprite
ld (bc),a     ; DE=écran
ldi           ; BC=mémoire
```

Compris ? Ok, quelques explications s'imposent. Les deux premières instructions effectuent la sauvegarde. Le LDI s'occupe de l'affichage, va incrémenter HL et DE... et va décrémenter BC ! Et c'est là que réside l'astuce. Oui, en faisant pointer BC à la fin de l'emplacement mémoire résérvé à la sauvegarde, le décor sera stocké à l'envers. Il nous suffira alors de le réafficher dans le bon sens lors de la restitution. Utilisez les registres secondaires pour les boucles si vous ne voulez pas les recopier. Si vous avez choisi d'éliminer l'étape de sauvegarde, notez que l'on peut effectuer également restitution et affichage grâce à cette méthode. Voir même les trois étapes avec la même routine en rusant un peu... A vous de creuser.

La fin du début

Ainsi s'achève ce premier cours sur la création de jeux. J'attends maintenant votre courrier (voir mon adresse en Rubrique X) pour que vous me fassiez part de vos critiques et conseils sur l'évolution de ce dossier. De même, si vous avez un problème ou une question, je me ferai un plaisir d'y répondre et de vous aider. Sur ce, bon code, et à bientôt dans le prochain numéro !

Hicks

```
;
; Routine d'affichage et restitution
;          de sprites
; Par Hicks pour Quasar CPC (04/10/99)
;

        org &4000
        nolist

; Listing 1 - Restitution classique

        ld hl,decor     ; HL=décor à restituer
        ld de,ecran     ; à l'adresse écran
        ld a,haut
rest1   ldi             ; autant de ldis que le
        ldi             ; sprite est large en
        ldi             ; octets, exemple pour
        ldi             ; 5 octets
        ldi
        ex hl,de        ; car DE=écran et le BC26
        call bc26       ; est fait pour HL=écran
        ex hl,de        ; hop DE=écran à nouveau
        dec a
        jp nz,rest1
        ret


; Listing 2 - L'autogénéré
;
; Appeler la routine générée avec
; hl=mémoire écran
; de=&800-larg-1
; bc=&c050

        ld hl,&2000     ; adr de la future routine
        ld de,sprite    ; de pointe sur le sprite
        ld c,haut
auto1   ld b,larg
auto2   ld a,(de)       ; prends octet sprite
        or a            ; si nul on ne l'affiche
        jr z,noaff      ; pas en sautant à noaff
        ld (hl),&36     ; code de ld HL,(nn)
        inc hl          ; ceci est donc le bout
        ld (hl),a       ; de prog qui code
        inc hl          ; l'affichage d'un octet
noaff   inc de          ; octet suivant du sprite
        ld (hl),&23     ; code de inc HL
        inc hl

        djnz auto2      ; larg=larg-1

        dec c           ; test si on passe a la
        jr z,fin        ; dernière ligne
; &bc26
        dec hl          ; si non, on dec HL pour
                        ; effacer le inc HL inutile
        ld (hl),&19     ; code de add HL,DE
        inc hl          ; puis celui de
        ld (hl),&30     ; jr nc,$+1 pour coder le
        inc hl          ; BC26 dans le buffer
        ld (hl),1       ; $+1: saute d'un octet si
        inc hl          ; le carry n'est pas mis
        ld (hl),&09     ; code de add HL,BC
        inc hl

        jp auto1        ; ligne suivante

fin     ld (hl),&c9     ; fin ! code du ret
        ret


; Listing 3 - Masquage au pixel

        ld hl,ecran
        ld de,masque
        di              ; coupe les inters car on
        exx             ; utilise les registres
        ld hl,sprite    ; secondaires
        ld e,haut
pixl2   exx             ; le paire de registres
        ld b,larg
pixl1   ld a,(de)       ; prend octet du masque
        and (hl)        ; masque AND ecran
        inc de
        inc hl
        exx             ; 2e paire de registres
        or (hl)         ; OR sprite
        inc hl
        exx             ; le paire de registres
        ld (hl),a       ; affiche résultat
        djnz pixl1      ; octet suivant
        call bc26
        exx             ; 2e paire de registres
        dec e
        jp nz,pixl2
        ei              ; autorise les inters
        ret

; Listing 4 - Le bitplan

        ld hl,ecran
        ld de,sprite
        ld a,haut
bitp1   ld c,a          ; sauvegarde la hauteur
        ld b,larg
bitp2   ld a,(de)       ; on effectue un OR entre
        or (hl)         ; le sprite et l'écran et
        ld (hl),a       ; grace au jeu d'encre on
        inc hl          ; a une précision au pixel
        inc de
        djnz bitp2
        ld a,c          ; restitue la hauteur
        call bc26       ; dans A car C est modifié
        dec a           ; par le BC26
        jp nz,bitp1
        ret


; Call &bc26

bc26    ld bc,&800-larg ; descends de ligne en
        add hl,bc       ; ligne sur le meme bloc
        ret nc          ; de caractère
        ld bc,&c000+80  ; si débordement alors
        add hl,bc       ; on passe au bloc suivant
        ret
```

Langage C

Vous en rêviez depuis des lustres ! Eh bien Quasar CPC l'a enfin fait ! Voici une rubrique consacrée au Langage C... mais pas n'importe lequel... le Langage C sur CPC et pour CPC !

En fait il existe plusieurs solutions pour programmer en C sur CPC mais celle qui m'a paru être la plus viable consiste à utiliser le Small C. Ce kit de développement, disponible en freeware sur de nombreux sites Internet et sur le CD-ROM de Genesis8, est normalement réservé au CP/M. Mais, grâce aux bibliothèques spécifiques au firmware développées par Juergen Weber, celui-ci nous permet désormais de créer des exécutables Amsdos. Attention toutefois, la compilation s'effectue toujours sous CP/M.

Le principe

Tout d'abord, sachez que tout le nessécaire pour utiliser le Small C est disponible sur le Quasar Disc 2 ainsi que tous les listings. Et avant de parler de Langage C proprement dit, nous allons plutôt voir comment fonctionne ce kit.

La compilation s'effectue en quatre phases. Tout d'abord, grâce à CC.COM, le source C est transformé en source Macro-Assembleur (.ASM) :

```
CC -M nomfich
```

Ensuite, on utilise ZMAC.COM, pour assembler ce source assembleur. On génère alors un fichier objet (.OBJ) qui est en quelque sorte un exécutable relogeable.

```
ZMAC nomfich=nomfich
```

Chaque .C est à présent transformé en .OBJ et il s'agit maintenant de faire l'édition de lien. C'est-à-dire que l'on va transformer ces .OBJ en executables. Chaque programme utilisant des routines provenant de diverses bibliothèques il va falloir inclure les .OBJ correspondants à celles-ci dans notre .COM :

```
ZLINK nomfich=cpciolib,nomfich,...
```

Oh ! Le petit blond à lunettes a réagit ! Et il a raison. Le fichier .OBJ principal n'est pas celui du fichier mais celui de la CPCIOLIB de Juergen Weber. En effet, notre programme n'est qu'un sous programme de cette bibliothèque qui nous permet de bidouiller une compilation pour Amsdos. Le source C de cette bibliothèque est présent sur le Quasar Disc 2 et vous pourrez si vous le désirez regarder ça de plus près... il y a pas mal d'assembleur en fait.

Vient ensuite la dernière étape : transformer notre .COM (qui est en fait un code firmware ne fonctionnant absolument pas sous CP/M) en .BIN tout ce qu'il y a de plus standard.

```
COMAM nomfich
```

Voilà, nous avons à présent un beau binaire qui se lance tout bêtement par un RUN"nomfich". Le seul problème est que le binaire ainsi obtenu est obligatoirement logé en &100 et qu'il ne permet aucun retour au système. Concernant l'allocation en &0100, il est en théorie possible de "linker" les .OBJ ailleurs mais en pratique, ZLINK étant buggé, ce n'est pas possible. Il est en revanche envisageable de réécrire un "linker" plus performant car le format des fichiers .OBJ est parfaitement renseigné dans les fichiers .DOC distribués avec le kit Small C.

En ce qui concerne le retour au système, je ne me suis pas penché sur la question pour le moment, mais je pense qu'en modifiant la CPCIOLIB on doit pouvoir corriger ça.

Le vif du sujet

Plutôt que de commencer dès cette fois-ci par un véritable cours sur le Langage C, je vais vous présenter les bases de fonctionnement...

Vous devez savoir que nous avons essentiellement deux types de fichiers : les .C (comme C :) et les .H (comme Header). Les premiers sont nos sources C à proprement parler et les seconds, qui sont la plupart du temps associés à des .C et des .OBJ, sont juste là pour déclarer les variables, routines, etc... de ces derniers. Ah ! Le petit blond à lunettes est perplexe. En fait, il est extrêmement rare qu'un fichier C se suffise à lui-même, la plupart du temps il utilise des

routines définies dans des bibliothèques externes... Et c'est là qu'interviennent les fameux "include".

Ceux-ci sont là pour indiquer quels programmes externes devront être pris en compte lors de la compilation. Un include peut bien évidemment être un fichier C lui-même, mais celà signifie qu'à chaque compilation il va être traité de même que notre fichier C principal... Voilà qui n'est pas très intéressant s'il s'agit d'une bibliothèque de 100Ko qui, de tout façon, n'est jamais modifiée. Eh bien c'est là qu'intervient notre .H, celui-ci contient la déclaration de toutes le routines du .OBJ associé et permet donc leur incorporation dans notre fichier principal ; et ce, seulement à l'édition de lien (ZLINK).

Vous l'aurez compris, le rôle d'un fichier .H est donc de permettre l'utilisation de fonctions externes dans notre programme principal et ce, en évitant une compilation complète systématique. Certain compilateurs C sont capables de générer aumatiquement des .H à partir de .C ; mais ce n'est malheureusement pas le cas de celui-ci... On est donc contraint de les créer à la main...

Dessin de R.VINCENT

Les bibliothèques du Small C

Outre la CPCIOLIB qui contient les principales références aux fonctions firmware, le Small C est accompagné de tout ce qu'il faut pour gérer l'affichage le calcul, etc... Contrairement à la CPCIOLIB, le source C de ces bibliothèques ne n'est pas disponible mais on dispose des .H et des .OBJ associés ce qui est amplement suffisant.

Je ne vais pas vous détailler ici le contenu de toutes ces bibliothèques car elles sont accompagnées d'un fichier .DOC très explicite - mais si vous le souhaitez vraiment je pourrai revenir dessus dans un prochain article. Pour ce premier contact avec le langage C sur CPC je vais à présent vous initier au Small C en précisant autant que possible les différences avec le C classique.

La structure des programmes

Un programme C est découpé en fonctions. Chacune de ces fonctions pouvant accepter un ou plusieurs paramètres mais pouvant également en renvoyer, directement ou via des pointeurs. Une de ces fonctions a un rôle un peu particulier puisqu'elle contient le programme principal : il s'agit de la fonction "main". Le protocole de déclaration des fonctions diffère toujours un peu d'un compilateur à l'autre, mais celui du Small C est un peu particulier :

```
NomDeFonction(liste) déclaration; { corps }
```

"liste" contient la liste des noms de variable servant de paramètres, et déclaration contient la déclaration de ces variables. Il n'est en effet pas possible de faire le typage des variables dans la déclaration de la fonction elle-même ! Enfin, ce n'est pas trop contraignant. En outre, lorsqu'une fonction ne nécéssite aucun paramètre, les parenthèses rester obligatoire mais il est inutile d'y glisser un "void" comme cela est souvent préconisé.

Le corps de la fonction est ensuite à insérer entre accolades, comme d'habitude en C. Si vous avez un clavier français et que vous utilisez un éditeur de texte mal (ou pas) configuré, vous devrez utiliser "é" pour "{" et "è" pour "}".

Déclaration des variables

La déclaration des variables est ici tout à fait standard avec respect des déclaration locales (dans les fonctions) et globales. En revanche, tous les types ne sont pas disponibles et les doubles pointeurs n'existent pas... Ah le petit blond à lunettes ne sait pas ce qu'est un pointeur. Je ne vais pas me lancer dans des explications détaillées dès cette première fois mais en résumé, un pointeur se déclare en faisant précéder la nom de la variable par un "*" et

désigne l'emplacement de la variable plutôt que la variable elle-même :

```
int a,b; /* Déclaration de deux entiers a et b */
int *p;  /* Déclaration d'un pointeur sur un
            entier */
```

Pour conclure avec les variables, lors de leur manipulation, sachez que "*var" désigne la contenu de la variable pointée par var et que "&var" désigne l'adresse où est stocké var... Enfin, nous reviendrons sur tous ces points la prochaine fois.

Les marqueurs

Comme ne cesse de le bêler le petit blond à lunettes, je ne vous ai pas encore expliqué ce que signifiaient les /* et */ - mais, bon, je pense qu'à part lui tout le monde aura compris qu'il s'agit des marqueurs début et de fin de remarque ; ça y est, je crois que je l'ai vexé, tant mieux il nous fichera la paix ! Donc, ces remarques peuvent être insérée n'importe où dans vos sources, sans aucune contrainte de taille.

Mais il y a bien plus intéressant que les remarques... Tout d'abord il y a le #include nomfich, qui permet d'intéger un .C ou un .H. Et puis il y a le fameux #define nomconst valeur, qui permet d'attribuer une valeur à une constante - pratiquement, lors de la compilation, tous les "nomconst" seront remplacés par la "valeur" correspondant, ni plus, ni moins. Et puis, je gardais le meilleurs pour la fin, il y a les #ASM et #ENDASM qui permettent tout simplement d'insérer directement du code assembleur dans vos sources C ; lors de la compilation tout ce qui se trouve entre ces deux marqueurs est ignoré et sera traité directement par le macro-assembleur ZMAC.

Les instructions

Je n'ai pas la place de tout vous détailler mais je pense que les listings qui suivent pourront vous aider. Il y a tout de même quelques détails importants à noter. Concernant la manipulation des variables, les attributions du type "a+=b" ne sont pas reconnues, il faudra donc utiliser "a=a+b". Mais, tout ce qui concerne les "++a" ou "a++" fonctionne, ouf !

Ensuite, au niveau des boucles, une seule structure est à notre disposition : le WHILE(condition) {...}. Mais en pratique ce n'est pas une très grosse contrainte. En effet, à l'aide des post/préincrémentations il est aisé de remplacer un FOR() par un WHILE() (voir exemple). Voilà, ce sont les deux plus grosses limitations en dehors des types de données retreints.

Les programmes d'exemple

Ci-contre et ci-dessous vous avez trois programmes d'exemple : deux .C et un .H. Le CPCADDON.C est le source C de la bibliothèque perso que j'ai mise en place pour compléter la CPCIOLIB et le CPCADDON.H est le fichier hearder à inclure dans vos programmes utilisant le fonctions de celle-ci.

Et puis le fichier principal, EXEMPLE.C, vous montrera quelques exemples d'utilisations des boucles, des fonctions et des pointeurs. Celui-ci utilise CPCADDON pour les fonctions LOCATE() et KEYPRESSED(), PRINTF1 pour PRINTF() (voir dans les .DOC pour la différence avec le PRINTF() de PRINTF2)... Mais il utilise aussi bien évidemment CPCIOLIB pour pouvoir générer un exécutable Amsdos et utiliser les fonctions MODE(), etc...

J'espère que vous parviendrez à vous débrouiller avec ces quelques listings qui, quoiqu'assez élémentaires mettent en jeu pas mal d'éléments du Langage C en général et du Small C en particulier. Si vous voulez éviter de les saisir, ceux-ci présents sur le Quasar Disc 2 ainsi qu'un makefile "exemple.sub" et les résultats de la compilation (CPCADDON.OBJ, EXEMPLE.BIN).

Derniers conseils

Avant de vous laisser il faut que je vous avertisse que si vous oubliez des accolades il se peut que le compilateur C bloque (un CRTL-C quitte) sans vous renvoyer de message d'erreur. Ensuite, les noms des fonctions et des variables sont tronqués à 8 caractères ce qui n'empêche pas d'utiliser des noms plus longs pour plus de lisibilité. Enfin, l'utilisation de la CPCADDON induit parfois un "Warning Duplicate Declaration" mais c'est sans conséquence car le compilateur fait tout de même ce qu'il faut. Bonne bidouille en Small C et à très bientôt pour un véritable cours de C.

OffseT

CPCADDON.H

```
/* Fichier Header pour la CPCADDON Library

   OffseT/Futurs - Avril 2000

*/

#define SET_CUR_TXT 47989
#define TST_KEY 47899

#asm
        GLOBAL @LOCATE
        GLOBAL @KEYPRESSED
#endasm
```

CPCADDON.C

```
/*

   CPCADDON par OffseT
   pour Quasar CPC 18 - Avril 2000

   Exemple de fonctions CPC en complément
   de la CPCIOLIB de Juergen Weber

   A utiliser avec le compilateur Small-C

compilation :

   cc -M cpcaddon
   zmac cpcaddon=cpcaddon

*/

#include cpciolib.h
#include cpcaddon.h


int regs[4];                    /* AF HL DE BC */

locate(x,y) int x,y;            /* Fonction LOCATE */
{
 regs[1]=(x*256)+y;
 oscall(SET_CUR_TXT,regs);
}

keypressed(a) int *a;           /* Fonction KEYPRESSED */
{
 regs[0]=0;
 oscall(TST_KEY,regs);
 *a=regs[0]
}
```

EXEMPLE.C

```
/*

(Tout) Petit programme d'exemple en C
par OffseT/Futurs pour Quasar CPC 18

Compilation (sous CP/M 3) :

   cc -M exemple
   zmac exemple=exemple
   zlink exemple=cpciolib,exemple,printf1,cpcaddon
   comam exemple

*/

#include cpciolib.h     /* Bibliothèque C pour CPC */
#include printf1.h      /* Bibliothèque Printf */
#include cpcaddon.h     /* Bibliothèque Quasar CPC */

noballe(x,y) int x,y;   /* Effaçage balle */
{
 locate(x,y);
 printf(" ");
}

balle(x,y) int x,y;     /* Affichage balle */
{
 locate(x,y);
 printf("o");
}

initvdu()               /* Initialisation écran */
{
 int i;                 /* Variable locale */

 mode(1);
 i=2;
 locate(i,1);

 while(i++<40)
 {
  printf("*");
 }
 i=2;
 locate(i,25);

 while(i++<40)
 {
  printf("*");
 }
 i=1;
 locate(1,i);

 while(i++<24)
 {
  locate(1,i);
  printf("*");
  locate(40,i);
  printf("*");
 }
}
```

```
main()                  /* Programme Principal */
{
 int x,y;               /* Déclaration variables */
 int incx,incy;
 int a;

 a=0;                   /* Initialisation variables */
 x=10;
 y=10;
 incx=1;
 incy=1;
 initvdu();

 while(a==0)            /* Boucle */
 {
  keypressed(&a);
  noballe(x,y);
  x=x+incx;
  y=y+incy;
  balle(x,y);
  if (x==39) { incx=-1; }
  if (x==2)  { incx=1;  }
  if (y==24) { incy=-1; }
  if (y==2)  { incy=1;  }
 }
 mode(2);               /* Fin */
}
```

On se retrouve à la fin de l'été pour
le numéro 19 !