Le Reporter No 8 (c)1995 F.Sanchez

Au Sommaire de ce Numéro

Initiations

Techniques

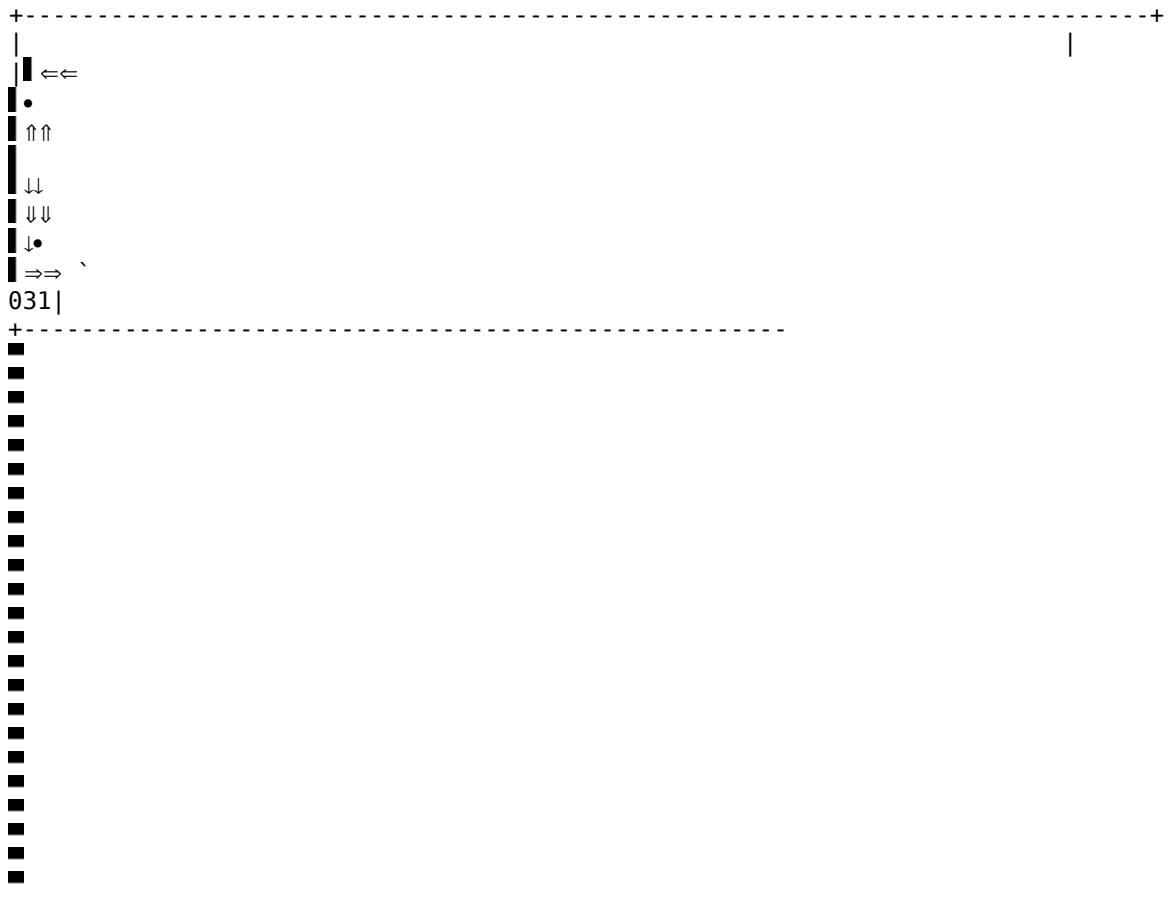
Ce que vous en pensez

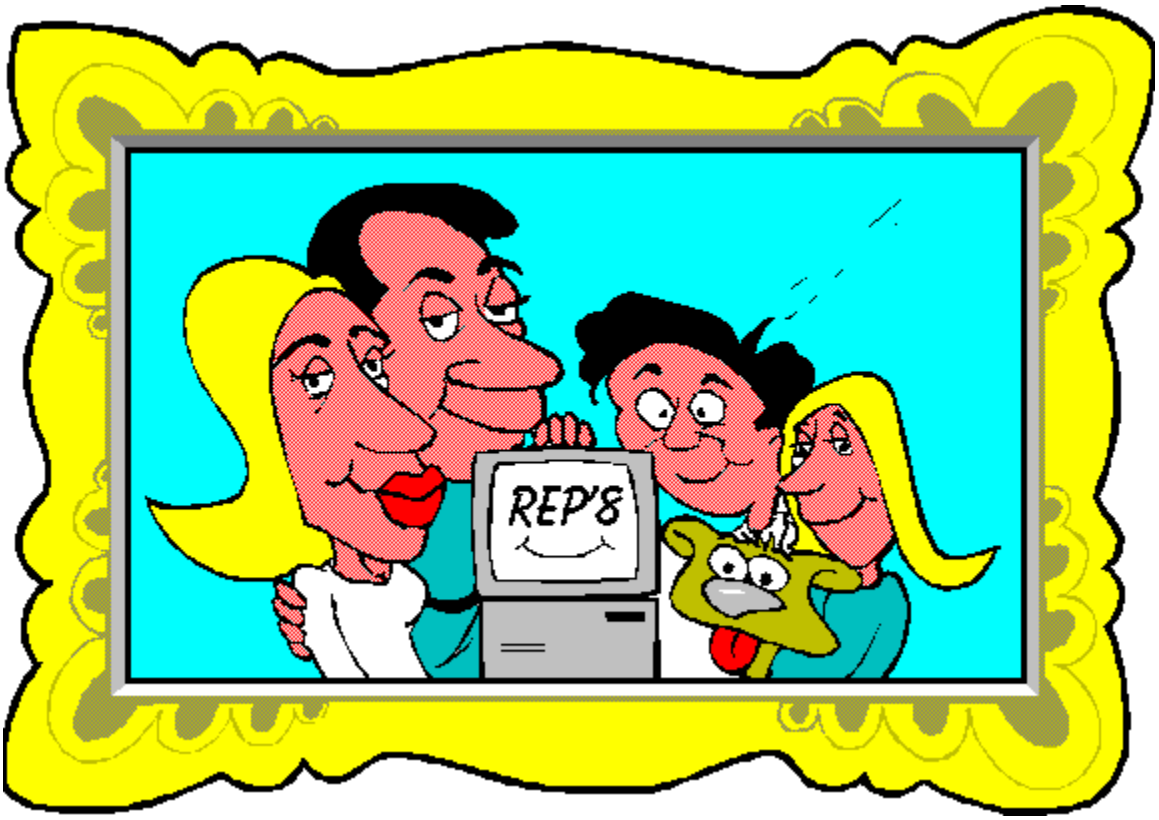
Assembly'95
L'intégrale
Runtime Error

Détente

Lamerland, nouveau parc d'attraction
Chroniques du Rezo V

"That's all folks!"




REP'S

A l'aide !!!

François CLAUSTRES (LEVIATHAN) [Sep 95]

A l'aide !!!

Le gestionnaire d'aide de Windows vous est sans doute familier...

Au pire, comme dans le Reporter DOS, **F1** vous donnera accès à l'aide sur l'hypertexte !

Néanmoins, voici un résumé rapide des commandes essentielles, pour ceux qui seraient rétifs à l'aide sur l'aide... Ou pour les lecteurs habitués de la version DOS du Reporter, pour lesquels nous donnons les équivalences WinRep' / REP' DOS.

D'abord la ligne de commandes :

◆ **Fichier - Imprimer la rubrique** permet d'imprimer la totalité de la rubrique en cours, y compris les graphiques. Equivalent de **ALT-P** et choix **Imprimante** depuis le REP'3.

◆ **Edition - Copier** permet d'accéder à un écran qui vous permettra de sélectionner le texte à copier de la rubrique courante. **Mais** cela ne copiera que le texte simple, et pas les caractères étendus (comme : ◆ ≡ ou ϒ) qui seront retranscrits en lettres plus ou moins incohérentes...
Equivalent de **ALT-P** et choix **Fichier** depuis le REP'3... Sauf qu'il faudra coller ce texte dans un éditeur (comme le Bloc Note) afin de le sauver.

◆ **Edition - Annotation** est l'une des commandes les plus intéressante : elle permet en effet d'annoter une rubrique d'un commentaire personnel... Seul inconvénient : on ne peut annoter une ligne particulière, car le **trombone** qui donne accès à cette note

se trouve toujours en début de rubrique.
On remarquera toutefois que cela n'abîme pas du tout le fichier HLP, dans la mesure ou WinHelp sauve ces notes dans un fichier à part.
Exclusivité du WinRep' par rapport au REP' DOS !!!

◆ **Signet - Définir** est **LA** commande à connaître : vous êtes en train de lire un article passionnant (mais long) sur les calendriers... et c'est bien sûr pile à ce moment là que vous devez partir et éteindre votre PC !
Au lieu de noter sur un bout de papier l'endroit où vous vous êtes arrêté, afin de ne pas passer ¼ d'heure à chercher le lendemain, il suffit de définir un signet... Et sur simple appel de celui-ci, vous retournerez exactement à l'endroit que vous aviez quitté la veille !
Exclusivité du WinRep' par rapport au REP' DOS !!!

◆ Le menu **Articles** vous permet de vous déplacer instantanément dans n'importe lequel des articles du numéro !

◆ **?** - **Toujours Visible** vous sera utile si vous êtes un vrai fana de Windows. Elle permet de garder le winrep' toujours au premier plan, même quand vous êtes dans une autre application. Cela vous offre ainsi la possibilité de suivre les articles tout en appliquant les exemples dans l'application concernée !!!
Il va sans dire que cette option elle aussi est une exlusivité WinRep' !

La ligne des boutons, ensuite :

◆ **Index** permet normalement de retourner à un sommaire général. C'est la règle suivie aussi pour le WinRep'... On peut la comparer au **F2** des REP'4 ou 5 sous DOS.

◆ **Rechercher** permet de... rechercher un thème donné. Equivalent au **CTRL-PgDn** des REP'4 ou 5 sous DOS.

◆ **Précédent** permet de revenir en arrière d'une commande. C'est l'équivalent de l'**ESC** dans les Reporters sous DOS.

◆ **Historique** permet d'accéder à une liste des commandes passées, et de reprendre à partir de n'importe laquelle de celles-ci. Equivalent au **CTRL-PgUp** des REP'4 ou 5 sous DOS.

◆ **Imprimer** permet d'imprimer directement le chapitre en cours.

◆ **<<** et **>>** permettent de naviguer d'une rubrique à l'autre, quand cette option est disponible. Equivalent à **SHIFT-Flèches** avec le REP'5 sous DOS.

◆ **Fermer** est un bouton spécifique au WinRep' : il permet de quitter rapidement le

WinRep' ! Equivalent de **ALT-X** depuis le REP'4.

Bonne(s) découverte(s) !

L'Edito du Rep 8

François Sanchez

Enfin ! Le reporter parait une fois de plus, pour votre plaisir...

Les vacances, ha les vacances !

La plage, le sable chaud, le soleil, les algues venant se déposer
sur les côtes emmenant avec elles des tas de choses pas toujours
recyclables... Les gorges d'ardèche. Les montagnes des pyrénnées. Les têtes nouvelles. Les têtes anciennes... Les uns nous apportant leur soleil, les autres faisant marcher le commerce.

Le départ des uns, le retour des autres...

Ha les vacances ! J'espère qu'elles furent bonnes pour vous, et qu'elles vous ont été bénéfiques.

Mais un bon coup de soleil, cela peut faire mal, très mal.

Je n'ai pas voulu prendre ce risque envers nos rédacteurs, imaginez un peu s'ils étaient partis ? Hop, pas de REP 8...

Et bien non, une fois de plus ils vous ont préparé des articles... Il y en a pour tous les goûts, comme d'habitude.

Ils étaient là bien au chaud, devant leurs ordinateurs à pianoter leur passion, à passionner leur piano...

Ha les vacances pour eux !

Allez, je vais pas être vache, pendant que vous tapez sur ESC, je les relache.

Mais où vont-ils tous ????? HELP !!!!!! Revenez très vite hein ??? Nos lecteurs attendent déjà le REP 9 ! Zalez pas croire que j'allais vous lacher plus de deux jours quand même non ? Gniark Gniark...

Au fait, cher lecteur, au cas où nos rédacteurs prendraient trop de vacances, ne vous génez pas pour nous ecrire quelque chose qu'il vous plairait de faire partager. Le format ASCII est amplement suf- fisant pour cela.

Je vous souhaite une très très bonne lecture avec ce numéro 8.

François SANCHEZ.

[ESC]

Les précédents Numéros

Le Reporter N° 7

 Le Reporter N° 6

 Le Reporter N° 5

 Le Reporter N° 4

 Le Reporter N° 3

La Rubrique des Rorettes
Réflexions
J'ai la flemme (Initiation générale à l'informatique I)
Initiation
Grosse Fatigue
Humour
Initiation à l'asm I
Initiation
Initiation au VGA III
Techniques/Hard/Programmation
Animations II
Techniques/Asm
Assembler en solitaire
Algorithmie appliquée/Asm
Traitement d'images III
Techniques/Pascal
TurboPascal pour débutants
Applications/Pascal
Convertisseur TGA->GIF
Techniques/Pascal/Graphisme
Algorithmique IV
Techniques/Initiation
Ligne brisées
Routines/Pascal
Trancher les chiffres
Routines/Pascal
Disques chromatiques
Techniques Visuelles
Approche Orientée Objet
Initiation/POO
Rotozoom
Techniques/Graphismes/Démos/Asm
Atelier Corel Draw I
Présentation/Trucs & Astuces
Lovecraft
Général S.F.
CyberPunk
Général/Culture
En Reseau
Histoires Vécues/Humour
Rubrique Aazienne
Histoires Vécues/Humour/Réflexions
Arroseur Arrosé (François Sanchez)
Interview
Démos PC
Démos
Pc Parfait (comment monter son PC en Kit)
Hard
CPC Emul (Logiciel)
Présentation
CVPerfect (Logiciel)
Présentation
Savoir (Logiciel)
Présentation
Luxart (Logiciel)
Présentation
Navidoc (Logiciel)
Présentation
Club Win (Services)
Présentation

 Le Reporter N° 2

Borland C++
Réponse Générale
Construction et animation d'objets 3D Réponse Générale
Projet Odysee (Ecriture d'un logiciel de modélisation)
Présentation
Algorithmie III
Techniques/Initiation
Résolution du problème des 8 reines en C Algorithmie appliquée
Interprête d'expressions II
Techniques/Pascal
Traitement d'images II
Techniques/Pascal
La Grande Saga de la 3D
Techniques/Graphismes
Spécifications du 386
Techniques/Hard
Animation I Code généré
Techniques/Asm
Initiation au VGA I
Techniques/Hard/Programmation
Programmation de TSR
Techniques/Asm
L'ordinateur et l'avion (Processus

d'initialisation d'un PC du BIOS à l'autoexec.bat) Hard
Rôle et règles de conception des documentations de Sharewares
Réflexions
Chronologies Parallèles (Petite histoire des grands noms de l'informatique)
Histoire/Références
Luc Rivière auteur d'Esope
Interview
Rubrique Démos
Démos
Gigantex RTC (Services)
Présentation
U.G.U.I. Bibliothèque Classe C++ (Logiciel) Présentation
Association Micro Contact's (Services)
Présentation
The Hand Of Fate
Test jeux
Rosendo
Histoires Vécues/Humour

 Le Reporter N° 1

Le Reporter N° 8, Septembre 1995

Rédacteur en Chef : François SANCHEZ AAZ,
Correcteur : Jean-Paul MARUEJOULS CYBERMAD,
Image d'intro : SUPAD xx,
Interfaçage Dos : Stéph' PINEAU STESSY,
Interfaçage Win : Charles LAHLOU SQUIRREL,
Rédaction : Cédric CELLIER RIXED,
François CLAUSTRES LEVIATHAN,
Christophe DARLOT A. ACCROC,
Philippe DEBREUCQ DIONYS,
Julien GILLE GILLEJ,
JCRxx,
Charles LAHLOU SQUIRREL,
Jean-Seb LEBARBIER WARRANT,
Jean-Paul MARUEJOULS CYBERMAD,
Emmanuel PIERRE SCYTHALE,
Stéph' PINEAU STESSY,
PSYCHESSxx,
Nicolas TORNERI TrIaX

Abonnez-vous, c'est gratuit !

Pour vous abonnez faites nous parvenir :

4 enveloppes pré-adressées et pré-affranchies à 4.40f
(Pour l'étranger joindre aux enveloppes l'équivalent en coupons réponses internationaux du coût d'envoi de 4 disquettes - Renseignement auprès de votre service des postes)
4 disquettes 3'1/2 1.44 MO
1 petit mot pour indiquer à partir de quel numéro vous souhaitez recevoir le Reporter dans votre boîte aux lettres dès sa parution.

Envoyez le tout à : François Sanchez,
14-940, rue Louis BEYDTS
33310 LORMONT

Attention ! Toute demande incomplète (disquettes 720ko, mauvais timbrage, pas d'adresse de retour...) sera ignorée.

Vous pouvez également depuis le N°6 du Reporter vous procurer les nouveaux numéros sur INTERNET à l'adresse :

ftp://teaser.fr/pub/indispensables.pc/le-reporter/

Anciens Numéros

Vous pouvez vous procurer les anciens numéros gratuitement en adressant une disquette 3"1/2 1.44 MO ainsi qu'une enveloppe pré-affranchie à 4.40Frs et pré-adressée par "Pack" voulu :

Pack 1 : REPORTER 1 + REPORTER 2 + REPORTER 3
Pack 2 : REPORTER 4 + REPORTER 5 + WINREP 0
Pack 3 : REPORTER 6 Version DOS et WINDOWS
Pack 4 : REPORTER 7 Version DOS et WINDOWS

Envoyez votre demande à :

François Sanchez,
14-940, rue Louis BEYDTS
33310 LORMONT

Contactez-nous/Rejoignez-nous !

Pour contactez un auteur d'article, sélectionnez son nom dans la liste ci-dessus pour obtenir ses coordonnées.

Pour nous faire part de vos réactions sur le Reporter en général, pour nous soumettre des articles écrivez à l'attention de François Sanchez (adresse ci-dessus) ou laissez un mot sur le serveur télématique* :

3614 TEASER b.à.l. REPORTER
ou 3614 TEASER Club LEREP
ou Email : aaz@email.teaser.fr

(* 0.36f/mn aux heures normales, et même tarif dégressif aux heures creuses qu'une communication téléphonique.)

(c) Copyright

Par Minitel sur le 3614 Code TEASER

Boite-Aux-Lettres : REPORTER.

ou Club Public : LEREP

Email : aaz@email.teaser.fr

SQUIRREL & WARRANT

Nicolas TORNERI
16, rue des Marroniers
62840 FLEURBAIX

Adresses Télématiques :

3614 TEASER b.à.l. TrIaX
3614 RTEL1   b.à.l. TrIaX

Emmanuel PIERRE
12, rue Joseph Dijon
75018 PARIS

Adresses Télématiques :

3614 RTEL1 b.à.l. SCYTHALE

Christine BAUDRY
14-940 rue Louis BEYDTS
33310 LORMONT

François SANCHEZ
14-940 rue Louis BEYDTS
33310 LORMONT

Adresses Télématiques :

3614 TEASER b.à.l AAZ

Email: aaz@email.teaser.fr

Cédric CELLIER
14 rue de la Libération
91480 QUINCY-SOUS-SENART

Adresses Télématiques :

3614 TEASER b.à.l RIXED

E-Mail : Cedric.Cellier@nuxes.frmug.fr.net
ou : cedric.cellier@nuxes.frmug.fr.net

Barthélémy BROLA
31-1202 rue Jean-Jacques Rousseau
33200 BORDEAUX-CAUDERAN

Benoît DELPLACE
"Chantevent"
02480 CUGNY

Christophe DARLOT
2, rue Bas des vignes
70000 QUINCEY

Adresses Télématiques :

3614 TEASER b.à.l. A. ACCROC

Philippe PALAU
112 rue du Chemin Vert
75011 PARIS

Luc RIVIERE
15 rue Henri DUNANT
29490 GUIPAVAS

Adresses Télématiques :

3614 TEASER b.à.l. LEGOLAS

Sébastien PISSAVY
14 rue de la Passerelle
15170 NEUSSARGUES

Adresses Télématiques :

3614 TEASER b.à.l. LIGHTMAN

Patrice DUMAS
Le village
07700 SAINT MARTIN D'ARDECHE

Adresses Télématiques :

3614 TEASER b.à.l. PULCO

Stéphane PINEAU
15 bis rue Bonnet
95400 ARNOUVILLE-LES-GONESSE

Adresses Télématiques :

3614 TEASER b.à.l. STESSY
3614 DP b.à.l. STESSY
3614 PICKUP b.à.l. STESSY
3614 CHEZ*STESSY

Jérôme STOLFO
Vieille Côte
Serre de CAZAUX
31800 SAINT-GAUDENS

Adresses Télématiques :

3614 TEASER b.à.l. ULTIMAN

Olivier PARISY
150 Avenue Jean Guiton
17000 LA ROCHELLE

Adresses Télématiques :

3614 TEASER b.à.l. HARDWARE

Jean-Paul MARUEJOULS
9 bd henri IV
34000 MONTPELLIER

Adresses Télématiques :

3614 TEASER b.à.l. CYBERMAD
3614 CNX b.à.l. CYBERMAD
3614 NOT b.à.l. CYBERMAD

E-Mail : abi@email.teaser.com

Arnaud LAUNAY
100, Avenue de la République
77450 ESBLY

Adresses Télématiques :

3614 TEASER b.à.l. LAUNAY

Pierre CHASSAING
53 rue Francis de Préssensé
44000 NANTES

Stéphane MARTY
La Martelle Bât G
845, Avenue de M. TESTE
34070 MONTPELLIER

Jérôme SPENLEHAUER
10 rue des Alouettes
68320 BISCHWIHR

Sébastien FILLEUL
Le Clos du PERCHE
13, Allée Jean GABIN
61300 L'AIGLE

Adresses Télématiques :

3614 TEASER b.à.l. SEBAFIL

Jean-Claude MEIER
17, rue de la Tuilerie
57420 FLEURY

Adresses Télématiques :

3614 TEASER b.à.l. JCMEIR

Sylvain LEBRETON
Kergolay en Moreac
56500 LOCMINE

François CLAUSTRES
Résidence de la Tour
12 allées Rés. Saint Mury
38240 MEYLAN

Adresses Télématiques :

3614 TEASER b.à.l. LEVIATHAN

Jean-Seb LEBARBIER
12 rue Malakoff
51200 EPERNAY

David DUCASSOU
Route de Monségur
40700 HAGETMAU

Florent RAMIERE
1 avenue de Paris
78000 VERSAILLES

Adresses Télématiques :

3614 TEASER b.à.l. MOGAR
E-Mail : ramier_f@epitat.fr

Philippe DEBREUCQ

Adresses Télématiques :

3614 TEASER b.à.l. DIONYS
E-Mail : debreucq@ens.insa-rennes.fr

Charles LAHLOU
128 av. du Maine
75014 PARIS

Adresses Télématiques :

3614 TEASER b.à.l. SQUIRREL
3614 RTEL1   b.à.l. SQUIRREL
3614 CNX   b.à.l. SQUIRREL

Jean-Pierre RAMBAUD
4 villa Parmentier
92120 MONTROUGE

Adresses Télématiques :

3614 TEASER b.à.l. REMBARRE

Anne M Beignatborde
La Herrane
32800 REANS

Michel BUKOVSKI
21, rue Henri Chevreau
75020 Paris

Adresses Télématiques :

3614 TEASER b.à.l. XOLO

Frédéric SEGURET
37, rue Angoulême
16100 COGNAC

Adresses Télématiques :

3614 TEASER b.à.l. SULTAN
3614 RTEL1   b.à.l. SULTAN

GILLE Julien
14 Grande Rue
54 170 ALLAIN

Adresses Télématiques :

B.à.L GILLE J. sur Teaser
Adresse 100657,1745 sur CompuServe
Adresse 100657,1745@compuserve.com sur Internet

Quoi d'neuf ?

Squirrel

Comme vous l'avez déjà remarqué, ce nouveau winrep' ressemble comme deux gouttes d'eau à ses petits frères !

Donc pas beaucoup de nouveautés cette fois, mais vous avez pu voir/verrez que l'accent à été mis sur les images, et désormais le Reporter possède sa propre icône personalisée quand vous le réduisez !

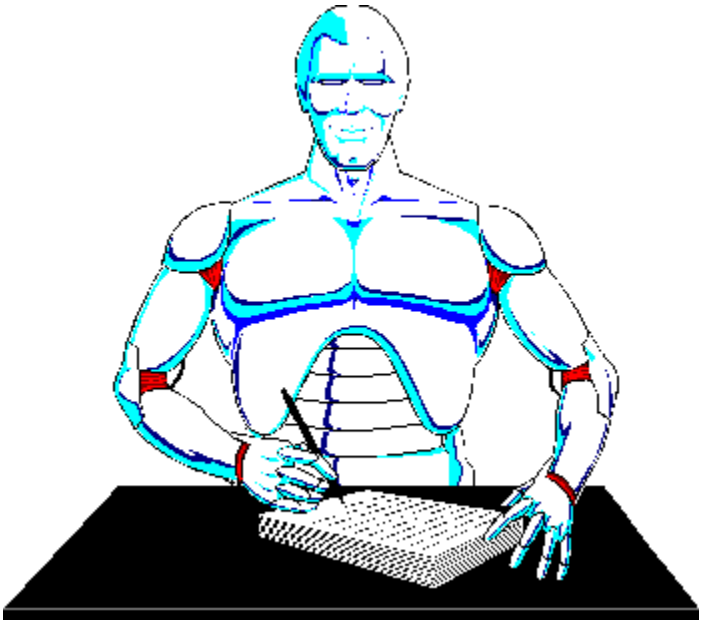
Un menu "Articles", dans la barre de menus fait son apparition. Il vous permet à tout moment de vous déplacer dans n'importe quel article, sans repasser par le sommaire. Bref, encore un peu plus de facilité de lecture !

La barre de boutons quant à elle a été comblée avec un bouton "imprimer".

Voilà ! Maintenant allez vite dévorer le Rep'8 !!!

Chaos Symbiotique 1

Chroniques du Rezo V [Cybermad]



Le ciel avait son habituelle couleur de déprime anxieuse, sous le plexiglas crasseux des dômes. Néo-Paris s'asphyxiait consciencieusement sous les vapeurs combinées des usines et des hydrocarbures.

Laura Graff se sentait déplacée dans cet amoncellement de rebuts du siècle précédant. Des carcasses de voitures à essence, d'avions à hélice ou de jets finissaient de rouiller, amalgamées au hasard en collines déchiquetées et rouillées. Mais qu'est-ce qu'il lui avait pris d'accepter un rendez-vous dans un coin aussi pourri, même pour un scoop de cette importance...

Laura contempla son reflet dans un bout de rétroviseur constellé de chiures de mouches : brune, des yeux bleus profonds, un visage ovale, des traits fins et délicats, que seules les commissures des lèvres venaient troubler, lui donnant un air tour à tour mutin ou inquiétant, selon son humeur. Elle portait un treillis spatial gris passe-partout et des rangers anti-grav. Cette tenue avait le double avantage de s'adapter parfaitement au lieu, et de masquer ses formes un peu trop généreuses pour ce trou à rat, où les pires déviants et détraquées nichaient par clans entiers.

Elle verifia sa check-list logicielle, histoire de tuer le temps. Une puce greffée sur son nerf optique retransmettait, à la demande, l'image qui se formait sur sa rétine vers un moniteur placé dans les bureaux de la Legamax, la compagnie de sécurité privée, où vers les studios de KAOS TV, la chaîne pour laquelle elle poireautait en ce moment, dans cet amas de fange nauséabonde. Il allait leur coûter cher, le résultat de son enquête !

De l'arrière du cadavre cannibalisé d'un ancien bus scolaire sortit un animal étrange. Elle crut tout d'abord avoir à faire à un chien mutant, résultat hasardeux de la sur-pollution et des fuites chimiques ou radioactives des zones industrielles périphériques. Mais quand il fut en pleine lumière, Laura reconnu un des anciens combattants du conflit Chinois. Les grossiers câbles sensitifs qui sortaient de ses tempes ne laissaient aucun doute. Ces liaisons neuronales étaient branchées directement sur le processeur de contrôle des avions de chasse ou de reconnaissance, laissant les mains libres pour l'utilisation des systèmes d'armement.

Apparament, c'était bien son contact. Elle ne savait de lui que son métier : ancien soldat, et son nom, Jack "Fool" Budsman. Mais il avait des informations à vendre. Des informations ultra-confidentielles, et surtout très sensibles : les preuves de la corruption quasi institutionnalisée du pouvoir politique, en fait les pantins fantôches des méga-corporations omnipotentes, mélange de maffia sicilienne et de zaïbatsu nippone...

Devant Laura, Jack "Fool" Budsman marqua un temps d'arrêt, comme s'il écoutait un lointain message ou une voie off. Il fixa la journaliste un long moment, et laissa tomber : "Un corrupteur vient d'être effacé ! Votre enquête va devenir de l'archéologie, si ça continue". Laura hésitait entre la répulsion, le dégoût ou la franche nausée. Si ce type avait eu un jour figure humaine, il n'en restait aucune trace aujourd'hui, que ce soit dans son corps ou dans son regard. Elle se sentait un peu dépassée, comme face à un androïde futuriste.

Jack tripotait sans cesse ses connexions neurales, pendant que son oeil droit tressautait en rythme ; toum...toum...tsii, toum...toum...tsii ! Son oeil gauche se cadra sur le visage de Laura : "une p'tite erreur de recâblage, ma prothèse occulaire droite est reliée à mon scanner radio". Laura avala difficilement sa salive, puis articula : "vous avez des infos à me vendre, je crois ? Des vraies infos sur les milieux d'affaires ?"

Jack repris un air plus sérieux : "Ouaip, des infos bien secrètes, du genre de celles qu'aiment les journalistes et le fisc !". Laura enchaîna rapidement : "Heu... comment pouvez-vous être au courant, vous ne me semblez pas très introduit dans les milieux financiers..."

"Vous fiez pas aux apparences", cracha Jack, "j'en sais plus sur vous que tous les services de polices réunis. Et même plus que vous même sans doutes...". Laura commençait à trouver la conversation pénible, mais elle savait qu'elle ne pourrait pas éviter d'aborder les sujets sensibles. Elle testa Jack : "Sur moi, vous ne connaissez rien, à part quelques informations publiques !".

Jack exhiba un énorme Smith & Wesson noir mat. Il releva le chien de l'arme et fixa Laura : "Ca, c'est ce que vous voulez croire, peut-être que ça vous rassure, mais je vais vous prouver que je ne suis pas un simple rigolo ! On va se redonner rendez-vous dans une semaine ici-même. En attendant, Ciao Bébé !" Jack pointa l'arme sur la tête de Laura, et appuya sur la détente. La jeune femme eut juste le temps d'apercevoir une flamme sortant du canon. Sa tête se désintégra dans une envolée d'os, de chair et de sang. Jack contempla le cadavre, et maugréa : "Encore une perte de temps, et pas moyen de faire autrement, hélas !"

*
* *

Dans un des innombrables bureaux du quartier des affaires, Molok Dzerdine trépignait. Deux heures ! Deux heures à attendre un coup de viosiophone qui ne venait pas. Pourtant, le cas de la journaliste devrait être réglé. Son contact lui avait assuré que ses hommes étaient les meilleurs, et qu'il pouvait considérer Laura Graff comme étant déjà décédée. Et toujours rien ! Pas un mot, pas un appel.

Molok jouait gros sur ce coup-là. Il avait mis dans la balance son influence politico-financière, sa fortune et sa vie. Si cette garce obtenait, et diffusait, la moindre information sur ses affaires, il pouvait dire adieu à cette débauche de luxe qui était son quotidien. Il décida de se passer les nerfs en faisant un tour dans le Rezo, et plaqua les trodes dorées à l'or fin sur les prises neurales de son crâne.

Dzerdine plongea dans le grand tourbillon d'information pure, tout son esprit tendu vers l'immensité de l'espace virtuel du Rezo. Il s'orienta vers une image brillante, couleur d'or et de miel : la "tour" de la SpaceNet Bank dont il était directeur et principal actionnaire. Il s'apprêtait à entrer dans la grande forme de lumière, quand une lueur d'un rouge ultra-vif attira son regard vers la droite. Il eut le temps d'apercevoir un faucon noir stylisé dont les yeux émettaient l'étrange lumière rouge, avant de s'engloutir dans une innommable noirceur, dans le gouffre de néant où sa vie s'achevait, brutalement, sans préavis.

*
* *

Laura reprit conscience dans son appartement. Elle s'extirpa de son cryo-bloc, un peu chancelante. Ce Jack "Fool" Budsman de malheur n'avait pas menti, il en savait sur elle plus que n'importe qui. Mais comment avait-il pu apprendre qu'elle était un "construct", une personnalité sauvegardée sur des biopuces qui contrôle à distance un corps synthétique (bien imité au demeurant) ?

Elle replaca dans la cuve un corps de rechange. Mais cette fois, elle allait mettre une "amazone" ; réflexes câblés, techniques de combats intégrées, plus deux ou trois modules de détection et de filature. La vie de construct avait un gros avantage, c'est qu'on pouvait se programmer le corps que l'on voulait ! Pour l'instant, il n'était pas possible de changer de sexe, mais ça viendrait sûrement... Les progrès sont toujours fulgurants dans ce domaine.

Elle enclencha la régénération et passa dans le salon. Un appel attendait sur son vidphone : Jack "Fool" Budsman avançait le rendez-vous au même endroit à demain matin. Elle vérifia la date ; trois jours qu'elle avait été "effacée" par Jack. Ses résurrections prenaient de plus en plus de temps. Elle passa un appel prioritaire au Professeur BlutGeld, pour lui signaler ces décalages. Puis elle se rendit chez Clara Iron. Clara était un phénomène humain intéressant, journaliste, terroriste occasionnelle, tête brûlée, et accessoirement meilleure amie de Laura.

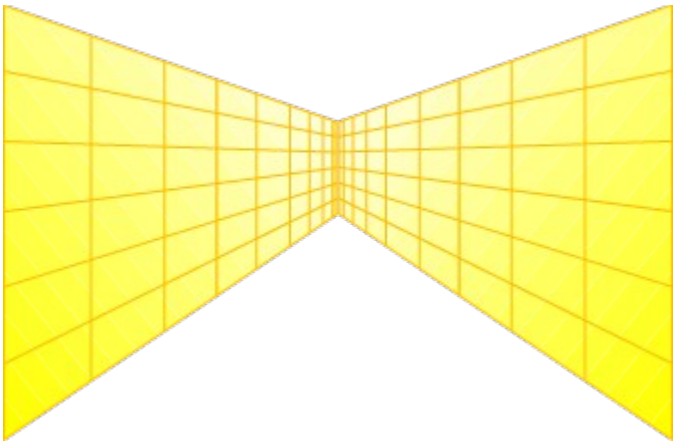
Clara la reçut vêtue en tout et pour tout d'un bonnet de bain vert fluo. Laura lui fit une bise claquante sur la joue et alla s'affaler dans un fauteuil. Clara lui apporta un plateau chargé de d'une bouteille de vodka et d'un verre, ainsi que d'une assiette de pilules, sans doute des amphés. Laura déclina l'offre, laissant Clara se servir un phénoménal verre d'alcool russe. Dès que celle-ci eut avalée les trois quarts de son verre, en une seule gorgée, elle attaqua : "Ma chérie, je crois que j'ai mis le doigt sur un problème important" ; Clara finit son verre, puis répondit : "Tiens donc ? Je suis étonnée qu'une personne aussi casanière que toi puisse avoir le moindre problème grave...". Laura laissa apparaître un sourire malgré elle : "Ne te moque pas, je suis tombée sur une salle affaire. Mon contact m'a tiré une balle dans la tête à notre première rencontre, et me redonne rendez-vous aujourd'hui !".

Clara abandonna son air taquin, et reprit le ton grave, mais posé, que Laura était une des seules personnes à connaître : "Donc il est au courant. Mais je croyais que le nombre de gens sachant que tu étais en fait un construct se comptaient sur les doigts d'une main ?". Laura soupira : "Et bien, j'ai l'impression qu'on les compte maintenant sur les pieds d'un mille-pattes !". Clara s'absorba dans un silence pesant. Elle lança : "Bon, du calme ma chérie... d'abord contacter Blutgeld et lui demander qui, selon lui, peut être au courant. Ensuite, je vais aller faire un p'tit tour de Rezo, histoire de récolter quelques rumeurs sur toi et ton Jack Foolman". Laura précisa : "Jack FOOL Budsman...".

(A SUIVRE)

Ray-tracing avec P.O.V. (I)

Cybermad [Jul/Aou 95]



Initiation à l'Image de Synthèse

Le Ray-Tracing avec PERSISTENCE OF VISION (POV 2.2)

Voici un moyen d'entrer de plein pied dans une des facettes les plus éblouissantes de l'Image de Synthèse : le **Ray-Tracing**, ou Lancé de Rayons. Pour illustrer ces propos, je vous propose d'utiliser un logiciel qui a fait ses preuves, et qui plus est dont l'utilisation est gratuite (**FreeWare**) : Persistence Of Vision (POV).

Je vous rassure tout de suite, il est inutile d'avoir fait Math Sup ou de connaître par coeur le Traité de Géométrie pour utiliser POV. A l'aide de quelques commandes très intuitives il est déjà possible de réaliser des choses assez impressionantes. Alors, si vous veniez faire un tour dans l'image virtuelle ?

Pour commencer, procurez-vous POV version 2.2 pour IBM PC et compatibles. Vous le trouverez facilement chez tout bon diffuseur de ShareWare, et sur la plupart des serveurs de téléchargement. Ce programme est en FreeWare, et peut être utilisé sans payer d'enregistrement. De plus, il existe un grand nombre de scènes et d'exemples. La version que j'utilise provient du serveur 3614 TEASER. Elle est dans le Répertoire RayTracing.

Mais avant tout, voici une petite liste du matériel dont vous aurez besoin. Vous le verrez, rien de bien compliqué :

Un PC 386 avec carte VGA au minimum. Bien sûr, un 486DX2-66 avec SVGA sera plus confortable, mais n'est pas indispensable. Les temps de calculs seront plus longs, c'est tout. Un coprocesseur mathématique sera toutefois le bienvenu si vous possédez un SX...

Du papier millimétré, un décimètre, un rapporteur, un compas et une équerre peuvent être nécessaires pour élaborer des images comprenant un grand nombre d'éléments, mais sont inutiles au début de cette initiation.

Vous voilà prêt à créer votre première image. Faisons un petit tour rapide du programme et de son utilisation.

POVRAY.EXE, le programme de calcul des images (on dit **rendre**une image) nécessite quelques paramètres pour fonctionner. Avant même de créer notre première scène, voyons comment lancer notre premier 'rendu' :

```
 POVRAY +Ifichier.pov +Ofichier.tga +d -v +x +w320 +h200
                                    |                |                |    |    |
|          |
                                    |                |                |    |    |
+-------- Hauteur et largeur
                                    |                |                |    |    |
de l'image. Ici en
                                    |                |                |    |    |
320x200
 Nom de la scène    -+               |                |    |    |
 à calculer                             |                |    |    |
+------------- Permet d'inter-
                                                      |                |    |
rompre le rendu
 Nom du fichier      ----------+                |    |
en cours par la
 Targa (TGA) de                                          |    |
frappe        d'une
 l'image finale                                          |    |
touche
                                                                          |    |
                                                                          |
+---------------- désactive                                              |
                                                                          |
l'affichage en
                                                                          |
mode texte.
                                                                          |
                                                                          |

+------------------- Active l'affichage

de l'image pendant

son rendu
```

Il ne nous reste qu'à créer notre premier fichier POV. Appelons-le **ESSAI1.POV** par exemple. Utilisez votre éditeur de texte favori (EDIT du DOS fera très bien l'affaire) pour créer ce fichier.

Dans POV, pour obtenir une image, il faut définir au minimum :

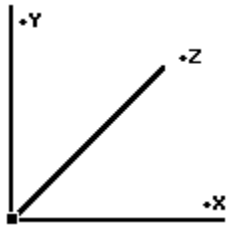
```
1 Caméra
1 Source de lumière
1 Objet
```

* Plaçons la caméra :

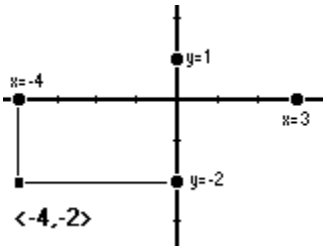
```
camera {
      location <0,0,-5>
      look_at <0,0,1>
}
```

Pour définir les paramètres d'un objet (camera, source de lumière, etc...) vous devez les intégrer dans des accolades {et }. Pour notre caméra, nous avons défini sa position (**location**) comme étant en x=0 y=0 z=-5.

J'en vois qui déjà fronce le sourcil, prêt à zapper ce malheureux article ! Allons, un peu de courage, voici comment on défini une position dans l'espace, dans l'univers de POV :



Pour simplifier, disons que **x** représente la largeur de l'écran, **y** la hauteur et **z**la profondeur. Considérons aussi que le point <**0**,**0**,**0**> est au centre de l'écran, une valeur négative pour x désigne la gauche et une valeur positive la droite, dans le cas de y, une valeur positive implique le haut, une négative le bas. Par exemple :



Pour la 3ème dimension (qui a dit la 4ème ?), c'est à peine plus compliqué car l'axe **z** représente une profondeur, ce qui, sur un écran en 2 dimensions est presque un paradoxe. Retenons, toujours en simplifiant à l'extrême, que si votre objet à une localisation **znégative**, il se rapproche de vous, et pour **z positif**, il s'éloigne vers le fond de l'écran. Ceci est une image imparfaite, mais suffisante pour ces débuts.

Donc, et si vous avez bien suivi (sinon remontez de quelques lignes), notre camera est positionnée au centre de l'écran (x=**0**et y=**0**) mais en retrait de **5** unités vers vous. C'est suffisament clair ? Continuons sur la lancée.

Nous avons, dans caméra, un autre paramètre, qui est **look_at**. Il sert tout simplement à indiquer vers où est pointée la caméra... facile, non ? ici, notre camera regarde droit devant elle, un peu en profondeur vers l'écran (x=**0**y=**0**comme la localisation, mais z=**1**).

On referme l'accolade et c'est fini pour la caméra.

* Une source de lumière sera bien utile, donc en voici une :

```
light_source { <10,10,-10> color red 1 green 1 blue 1 }
```

Devant cette ligne assez ésotérique, une explication s'impose, non ? la voici...

Le mot **light_source** indique une source de lumière (Ah Lapalisse !), c'est une traduction littérale. Toujours dans les **{**et **}** nous allons indiquer à POV les paramètres de cette source de lumière. Tout d'abord, la localisation (vous avez remarqué, chaque coordonnée est définie entre **<**et **>**) : Nous plaçons la lumière à 10 unités à droite, 10 en auteur et 10 en retrait vers nous.

Ensuite, nous précisons la couleur de la lumière. Il existe deux moyens de définir une couleur : la première en indiquant, pour chaque composante **RGB** (**Rouge**/**Vert**/**Bleu**) la quantité de chacune. C'est la méthode la moins facile. L'autre solution consiste à inclure le fichier **COLORS.INC** qui est livré avec POV. Ce fichier contient plusieurs couleurs prédéfinies que vous pouvez réutiliser. Nous allons légèrement modifier notre fichier ESSAI1.POV de manière à utiliser COLORS.INC :

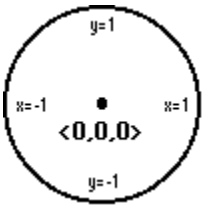
```
#include "colors.inc"

camera {
      location <0,0,-5>
      look_at <0,0,1>
}

light_source {
      <10,10,-10>
      color White
}
```

Si vous jetez un oeil à **COLORS.INC**, vous verrez que **White**est défini avec pour valeur : red 1 green 1 blue 1, ce qui nous ramène à la première version, le temps de recherche en moins (dans le cas d'autres couleurs). Nous verrons plus loin l'utilisation de la commande **#include** dans le détail.

* «On a la lumière, on a la télé, on peut y aller». Manque plus que l'objet. POV dispose de plusieurs objets prédéfinis d'origine, et de quelques autres dans les fichiers **SHAPES.INC**et **SHAPES_Q.INC**. Mais dans notre cas, on va utiliser le plus simple d'entre eux : la sphère. Ce qui donne...

```
sphere {
     <0,0,0>,1
     pigment { Red }
}
```

Vous aurez reconnu la localisation de la sphere, dont le centre est au point <**0**,**0**,**0**>. Le chiffre **1** indique le rayon de notre sphère. Vu de face notre sphère occupe l'espace suivant :



Ne confondez pas rayon et diamètre... vous auriez des surprises. Enfin, nous donnons une couleur à notre sphère. Un nouveau mot-clef est utilisé, il s'agit de **pigment { }**. Entre les accolades, on mettra la couleur. Ici on

a pris du Rouge, qui est en fait **Red** dans **COLORS.INC**. Attention aux majuscules/minuscules dans vos scènes, POV est "case sensitive". Dans notre scène, red et Red sont deux choses différentes. Le premier indique la composante rouge d'une couleur, le second la couleur Rouge prédéfinie dans **COLORS.INC** (Red = color red 1 green 0 blue 0). On referme l'accolade et on envoie le rendu de notre scène.

Si vous jetez un oeil au fichier **ESSAI1.POV** qui est quelque part dans les annexes de ce numéro, vous constaterez qu'il y a deux autres mots-clefs qui n'ont pas encore été vus ici :

/* et ***/** sont des marqueurs qui permettent d'inclure des commentaires associés à une scène. On place généralement une description du fichier ainsi que ce qu'il représente, et éventuellement plein d'autres détails comme le temps de calcul de la scène dans une dimension donnée et avec un processeur particulier, le nom de l'auteur etc... en tout début de fichier grâce à ces mots-clefs. Attention, pensez à commencer avec **/*** et à finir votre commentaire avec ***/**. Tout ce qui est entre sera ignoré par POVRAY.

// est aussi un mot-clef qui permet de placer des commentaires dans une scène, mais il indique à POVRAY d'ignorer ce qui suit, et ce jusqu'à la fin de la ligne. Voici quelques exemples :

```
               /* Fichier : Descriptif de scène pour POV 2.0           -+
                     Titre      : RELATIVITY
| Toute cette
                     Auteur     : Albert Einstein
| partie    est
                     Date        : 12/04/2006
| ignorée par
                     Durée      : 04'12" sur 486DX-33 en 640*480                |
POVRAY.
                     Droits     : Versé dans le Domaine Public                   |
|
                     Commentaire : Essayez de modifier les textures      |
                                        de BOIS ainsi que la localisation
|
                                        des sources lumineuses pour un
|
                                        meilleur effet
|

               */
-+

               #include "colors.inc"
               #include "textures.inc"
               #include "formes.inc"

               // Positionner la Caméra ---------------------------- Ligne ignorée
               camera {
                     location <0,10,-10>
                     look_at <1,1,-1>
               }

               object { Face_Avant } // pris dans FORMES.INC
             ...
+-------------------------- Fin de ligne
ignorée
```

Pour lancer le rendu, quittez l'éditeur et lancez **POVRAY.EXE** avec les paramètres suivants :

```
POVRAY +Iessai1.pov +Oessai1.tga +w320 +h200 +d -v +x
```

NOTE :

Vous pouvez remplacer la partie **+w320 +h200** par **+w640 +h480**. Ceci vous donnera une image de meilleure qualité (si vous avez une carte vidéo en SVGA uniquement) mais les temps de calcul seront plus longs (x6 environ). A réserver à des processeurs rapides !

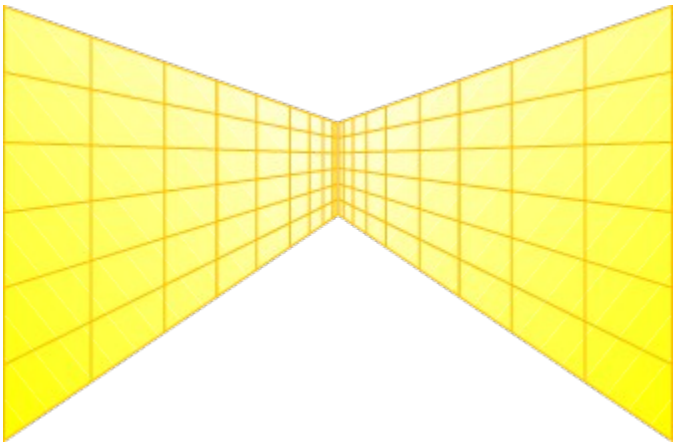
Maintenant que voilà réalisée notre première image, nous avons vu les fonctions de base de POV. La prochaine fois, on attaque plus sérieusement avec les autres objets, et la façon de les assembler pour créer des formes plus complexes, et on jetera un oeil à la notion de **TEXTURE** (ou comment faire une sphère en marbre, en bois, etc...) !

See you soon,

JiPé

Ray-tracing avec P.O.V. (II)

Cybermad [Jul/Aou 95]



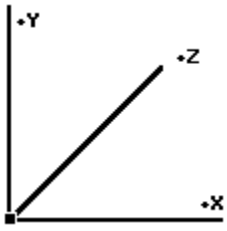
DEUXIEME PARTIE : les mots-clés "camera" et "light_source"

Cette seconde partie de l'initiation à POV 2.2 va nous permettre de régler la caméra et les différentes sources lumineuses possibles pour rendre une scène.

1) "camera"

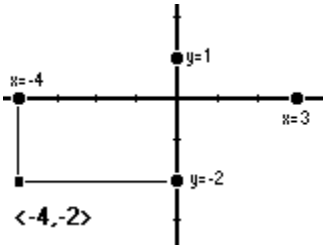
Nous avons utilisée une caméra dans le fichier **ESSAI1.POV**, mais sous sa forme la plus simple, sans lui donner le moindre élément de réglage autre que sa position et l'endroit où elle est pointée. POV 2 permet pourtant de faire pas mal de choses intéressantes en faisant juste varier les données de "**camera**".

Il est primordial de connaître le système de repérage de POV, selon les trois axes **x**, **y**et **z**. Pour vous éviter la fastidieuse tâche d'aller regarder plus haut , je vous explique à nouveau le système de coordonnées de POV :



Pour simplifier, disons que **x** représente la largeur de l'écran, **y** la hauteur et **z** la profondeur. Considérons aussi que le point <**0**,**0**,**0**> est au centre de l'écran, une valeur négative pour x désigne la gauche et une valeur positive

la droite, dans le cas de y, une valeur positive implique le haut, une négative le bas. Par exemple :



Pour la 3ème dimension (qui a dit la 5ème ?), c'est à peine plus compliqué car l'axe **z** représente une profondeur, ce qui, sur un écran en 2 dimensions est presque un paradoxe. Retenons, toujours en simplifiant à l'extrême, que si votre objet à une localisation **znégative**, il se rapproche de vous, et pour **z positif**, il s'éloigne vers le fond de l'écran. Ceci est une image imparfaite, mais suffisante pour ces débuts.

A] MOT-CLEF : "location < >"

Comme je vous l'expliquais dans le numéro précédant, ce mot-clef indique à POV ou se situe la caméra dans l'espace. Par exemple :

```
camera {
      location <5,10,-8>
      ...
```

place la caméra à 5 unités vers la droite, 10 vers le haut et 8 en retrait (vers nous).

B] MOT-CLEF : "sky < >"

Dans le système de coordonnées de POV, la direction du haut est donnée par l'axe des y, par une valeur positive. Toutefois, vous pouvez vouloir que votre caméra soit disposée différemment. Par exemple créer une vue depuis un avion en train de faire un tonneau, ou un piqué. Dans ce cas, on réorientera la caméra grâce à "**sky <x,y,z>**".

La valeur par défaut de "**sky**"est <**0**,**1**,**0**> (hauteur alignée sur l'axe des y). Si vous voulez créez une scène vue à l'envers (sol en haut et terre en bas) utilisez le vecteur "**sky <0,-1,0>**". Si vous voulez une scène penchée de 45 degrés à gauche, mettez "**sky <0,0.5,0.5>**", etc... NB : "**sky < >**" doit être placé avant "**look_at**".

C] MOT-CLEF : "direction < >"

Avec "**direction**", nous allons pouvoir régler l'objectif de notre caméra, c'est à dire utiliser un grand angle ou au contraire un zoom. Toutefois, utilisez ce mot-clef avec prudence : vous pourriez obtenir des scènes avec des effets bizarres de distorsion, surtout sur les bords.

La valeur par défaut de "**direction**" est <**0**,**0**,**1**>. Ici aussi, tant que la caméra est face à la scène, laissez x et y à zéro, sauf si vous comptez créer un effet vraiment spécial...

Voici quelques exemples de réglages de l'objectif. A vous de modifier la valeur de z jusqu'à obtenir le vecteur le

plus satisfaisant (Rappel : <x,y,z> est appelé vecteur) :

```
// Grand Angle
camera {
      location    <3,5,-10>
      direction <0,0,1>
      look_at      <0,2,1>
}

// Zoom (téléobjectif)
camera {
      location <3,5,-10>
      direction <0,0,8>
      look_at    <0,2,1>
}
```

Vous aurez sans doute remarqué qu'une petite valeur de z (une courte direction) donne un grand angle, et plus cette valeur augmente, plus on utilise un zoom lointain. Toutefois, faites très attention si vous utilisez une valeur de z inférieure à 1 (0.7, 0.95, 0.3, 0.15, etc...) car là, les déformations peuvent être énormes. Comme pour "**sky**", "**direction**" doit être placé avant "**look_at**".

D] MOT-CLEF : "up < >" & "right < >" (ratio d'aspect)

Les deux valeurs dans "**up <x,y,z>**" et "**right <x,y,z>**" donnent le rapport entre la hauteur de l'image finale et sa largeur. Ces valeurs par défaut sont "**up <0,1,0>**" et "**right <1.33,0,0>**", ce qui donne un rapport à peu près proportionnel à l'écran, de 4 de haut pour 3 de large. Pour les puristes, la largeur est égale à 4/3 de la hauteur...

Ces deux valeurs doivent correspondrent aux paramètres **+W**et **+H**que vous indiquez pour définir la résolution de l'image en lançant le rendu. Si nous prennons comme base une taille de **640**par **480**, nous pouvons vérifier que **640/480 = 4/3**, notre ratio est correct. Pour une résolution de **320x200**, il faudra un ratio de **320/200**, soit **8/5**. nous sélectionneront donc :

```
camera {
      location <0,0,-8>
      up <0,1,0>
      right <8/5,0,0>      // ou
<1.6,0,0>
      ...
```

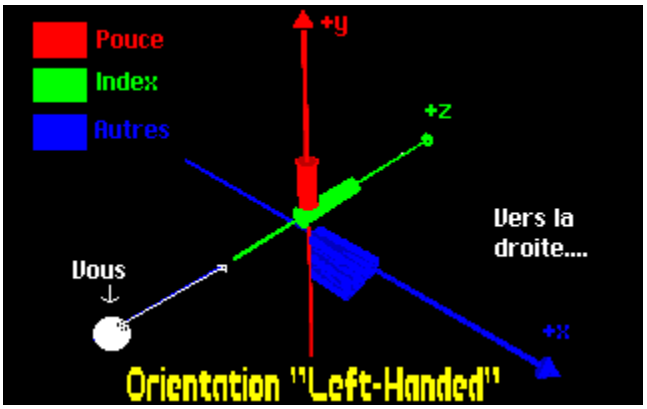
Le vecteur "**right**" sert aussi à définir le système de coordonnées. Nous avons revu plus haut que y indiquait la hauteur, x la largeur et z la profondeur. Ceci n'est vrai que pour le système par défaut, dit à main gauche (ou left-handed en Anglais). Vous pouvez utiliser votre main gauche pour matérialiser les trois axes :

- Levez votre main gauche, paume dirigée vers la droite ;

- Tendez le pouce vers le haut, votre index pointant droit devant ;

- repliez les trois autres doigts vers la droite, de manière à former un angle droit avec l'index.

Si vous gardez la pose, votre pouce indique l'axe des y, votre index l'axe des z et vos trois autres doigts l'axe des x, tous en positif.



Certains logiciels utilisent un système différent, où l'axe des z représente la hauteur, et l'axe des y la profondeur (LUXARTnotamment). Si vous entreprennez de "traduire" une scène de LUXART, plutôt que de recalculer tous les vecteurs, indiquez dans les paramètres de la caméra, par exemple :

```
// modifs adaptée à LUXART (vecteur <x,y,z>)
camera {
      location <0,0,-8>
      direction <0,1,0> // la profondeur est sur y
      up <0,0,1>                // la hauteur sur z
      right <1.33,0,0>    // la largeur ne change pas
      look_at <0,0,2>
}
```

Faites attention que les trois axes soient bien perpendiculaires les uns aux autres, sinon vous allez déformer votre image bizarrement... Ici aussi "**up**" et "**right**" doivent être indiqués impérativement avant "**look_at**".

E] MOT-CLEF : "look_at < >"

Cette fois, on indique où la caméra regarde :

```
camera {
      location <5,10,-8>
      look_at <0,0,2>
      ...
```

Toujours selon le même principe, x=0, y=0 et z=2. Une petite astuce, pour obtenir un meilleur effet de profondeur, évitez d'indiquer une valeur négative ou nulle pour z, du moins tant que la caméra est en face de la scène à rendre ou presque (ce qui est le cas ici !). le vecteur "**look_at**" doit toujours être indiqué le dernier.

F] VALEURS PAR DEFAUT

Si vous ne précisez rien, POV attribue les valeurs suivantes aux divers vecteurs :

```
- direction <0,0,1>
- up                <0,1,0>
- right          <1.33,0,0>
```

2) "light_source"

A] GENERALITES

Avec "**light_source**", littéralement source de lumière, nous allons pouvoir définir différents types d'éclairages, tels qu'une rangée de lampes, un spot, un objet lumineux (émettant une lumière), etc...

Comme pour "**camera**", nous allons passer en revue les différents mots-clefs et vecteurs qui peuvent modifier la source lumineuse. La description minimale pour "**light_source**"est :

```
                light_source { <x,y,z> color red a green b blue c }
ou :
                light_source { <x,y,z> color rgb <a,b,c> }
ou :
                light_source { <x,y,z> color NomDeCouleur }
```

Nous ne reviendrons pas sur la localisation, indiquée par le vecteur <**x**,**y**,**z**>. Par contre, il y a deux façons de définir une couleur pour la source de lumière. Soit on utilise une couleur prédéfinie, disponible dans le fichier **COLORS.INC**fournit avec POV, soit on indique la valeur de chaque composante rouge, verte et bleue d'une couleur (**RGB**). Dans ce dernier cas, deux syntaxes sont autorisées. Voici quelques exemples :

```
light_source { <0,20,-20> color red .15 green .95 blue .15 }
light_source { <10,10,-10> color rgb <.15,.95,.15> }

light_source { <10,0,-20> color red 1 green 0 blue 0 }
light_source { <10,0,-20> color red 1 }
```

Vous remarquerez que si une composante est à zéro, on peut l'omettre. A vous de trouver les bons paramètres, et les bons mélanges. Mais vous pouvez aussi utiliser toute une palette de couleurs prédéfinies dans le fichier **COLORS.INC**. Dans ce cas, il faut juste rajouter, en début du fichier de description de scène, la ligne :

```
#include "colors.inc"
```

Ainsi, POV gardera en mémoire toutes les composantes de chacun des noms de couleurs prédéfinis. Jetez un coup d'oeil à **COLORS.INC**avec votre éditeur de texte. Vous verrez quelque chose du genre :

```
// Many pre-defined colors for use in scene files.
// Also includes constants for Map Types and Interpolation Types
// for use with image_map, bump_map, and material_map
...
// These grays are useful for fine-tuning lighting color values
// and for other areas where subtle variations of grays are needed.
#declare Gray05 = color red 0.05 green 0.05 blue 0.05
#declare Gray10 = color red 0.10 green 0.10 blue 0.10
#declare Gray15 = color red 0.15 green 0.15 blue 0.15
...
#declare Clear = color red 1.0 green 1.0 blue 1.0 filter 1.0
#declare White = color red 1.0 green 1.0 blue 1.0
#declare Red = color red 1.0            // ATTENTION +      Red & red
+
#declare Green = color green 1.0    //    NE    PAS    | Green & green |
#declare Blue = color blue 1.0        // CONFONDRE +    Blue & blue    +
#declare Yellow = color red 1.0 green 1.0
#declare Cyan = color blue 1.0 green 1.0
...
#declare DarkTan = color red 0.59 green 0.41 blue 0.31
#declare GreenCopper = color red 0.32 green 0.49 blue 0.46
#declare DkGreenCopper = color red 0.29 green 0.46 blue 0.43
#declare DustyRose = color red 0.52 green 0.39 blue 0.39
#declare HuntersGreen = color red 0.13 green 0.37 blue 0.31
#declare Scarlet = color red 0.55 green 0.09 blue 0.09
...
```

Faites attention à ne pas confondre **Red**, couleur prédéfinie, avec **red**, composante de la couleur. Les Majuscules et minuscules doivent être absolument respectées dans POV. Idem pour **Blue**et **Green**. Quelques exemples valides :

```
#include "colors.inc"

camera {
      ...
}

light_source { <10,10,10> color White }
light_source { <10,10,10> color DarkTan }
light_source { <10,10,10> color Gray15 }
...
```

Vous pouvez définir plusieurs sources de lumière dans une même scène, mais plus il y en aura, plus long sera le temps de calcul de l'image.

B] MOT-CLEF : "spotlight"

Vous pouvez aussi créer une lumière du genre de celle d'un projecteur, ou d'un spot. Pour celà, on utilise le mot-clef "**spotlight**" avec tous ses paramètres :

```
light_source { <x,y,z>
      color NomDeCouleur
      spotlight
      point_at <x',y',z'>
      radius a
      falloff b
      tightness c
}
```

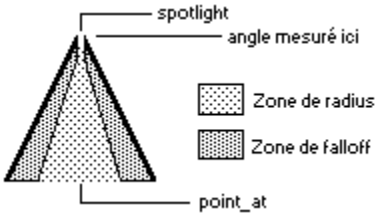
"**point_at**" le vecteur "**point_at**" s'utilise de la même façon que "**look_at**" pour une caméra. Il indique le point sur lequel est braqué le spot.

"**radius**" Ce mot-clef indique l'angle du projecteur en degrés. Dans cette angle la lumière est à pleine intensité. "**radius**" peut prendre une valeur comprise entre **1**et **180**.

"**falloff**" Ici, nous indiquons à POV sur quel angle la luminosité va-t-elle en décroissant. "falloff" doit être supérieur à "radius", et peut aussi être compris entre 1 et 180.

"**tightness**" Ce dernier mot-clef indique à quelle rapidité passe-t-on de la lumière à l'obscurité dans l'intervalle compris entre "radius" et "falloff". Plus la valeur (de**1**à **100**) est élevée, et plus crue sera la transition. une petite valeur donnera des ombres douces autour du pinceau lumineux. la valeur par défaut est **10**.

Un petit schéma aidera sans doute à mieux comprendre tout ça :



CJ MOT-CLEF : "area_light"

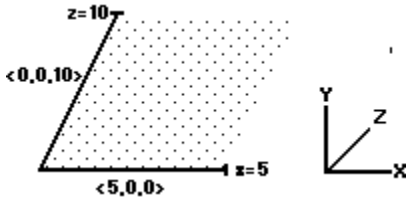
En général, une source de lumière dans POV n'émet qu'à partir d'un point unique. De ce fait, les ombres sont très vives, très marquées. Il est possible de simuler un autre type d'éclairage, comme un néon, ou une rangée de projecteurs. On utilise pour celà "**area_light**". L'effet principal est qu'il permet d'obtenir des ombres très douces. La syntaxe est :

```
light_source {
      <x,y,z> color NomDeCouleur
      area_light <a1,b1,c1>, <a2,b2,c2>, d1, d2
      adaptive
      jitter
      [éventuellement spotlight]
}
```

La localisation est indiquée comme pour une "**light_source**" classique, ainsi que la couleur. Les vecteurs <a1,b1,c1> et <a2,b2,c2> définissent un rectangle, dans lequel sont incluses les sources de lumière créant notre "**area_light**". Chaque vecteur indique un des points opposés du rectangle.

Les valeurs **d1** et **d2** indiquent le nombre de points lumineux présent le long de chaque vecteur. Par exemple :

area_light <5,0,0>,<0,0,10>,5,6 donnera :



Le mot-clef "**adaptive**" permet de réduire le nombre de rayons que devra tester POV. Plus sa valeur est élevée, et de meilleure qualité est l'ombre portée, mais plus long sera le temps de calcul. On n'a rien sans rien ! "**adaptive**" permet de faire un calcul approximatif moins gourmand en durée. Par défaut "**adaptive**" est à 1.

Enfin, le mot-clé "**jitter**" répartit aléatoirement les points lumineux sur la surface de cet "**area_light**". Il ne change rien au nombre de points lumineux et aux dimensions de la zone. Mais cette répartition étant totalement aléatoire, n'utilisez jamais jitter pour une animation.

D] MOT-CLEF : "looks_like"

Si vous voulez rendre la source lumineuse apparente, vous pouvez lui donner une forme, grâce à "**looks_like**". la syntaxe est alors :

```
light_source {
      <x,y,z> color NomDeCouleur
      looks_like { sphere { <u,v,w>,r texture { NomDeTexture }
}

light_source {
      <100,200,-200> color Red
      looks_like { sphere { <0,0,0>,2 texture { Glass } }
}
```

Une petite explication s'impose, non ? Avec cette disposition, la source de lumière aura la forme d'une sphère de deux unités de rayon, mais sera localisée en <1000,200,-200>. Nous verrons plus tard ce qu'est une "**texture**". On peut remplacer "**sphere**" par tout autre objet reconnu par POV.

Enfin, si vous souhaitez que l'objet rattaché à la source lumineuse bloque en partie la lumière émise (une lampe à abat-jour par exemple, n'utilisez pas "**looks_like**", mais le mot clé "**union**" rattachant l'objet à la source lumineuse. Mais ceci est une autre histoire...

Au prochain épisode, on ira faire un tour du côté de "**pigment {}**" et des divers **objets**disponibles dans POV.

See you soon,

JiPé

Init. aux Pointeurs en TurboPascal

Arthur Accroc

INITIATION AUX POINTEURS EN TURBO PASCAL

CONNAISSANCES NECESSAIRES : types ARRAY, RECORD, et tout le B-A-BA
de la programmation (boucles, tests...)

Le but de cet article est d'arriver assez rapidement à une bonne compréhension de ce qu'est un pointeur pour ceux qui n'en ont jamais entendu parler (ceux qui en ont entendu parler mais qui ne savent pas s'en servir peuvent également rester) et d'avoir un petit aperçu de ce que l'on peut en faire... Que les spécialistes me pardonnent les petites imprécisions que je serai parfois amené à faire afin de gagner du temps (les spécialistes n'ont d'ailleurs rien à faire ici vu que c'est marqué INITIATION à l'entrée ;-}).

Voici donc en gros ce que l'on va voir dans la suite de l'article :

Généralités sur les pointeurs

Arthur Accroc

I) GENERALITES SUR LES POINTEURS

PREAMBULE :

Lorsqu'on déclare dans un programme en pascal une variable, c'est pour permettre au compilateur de réserver de la place en mémoire pour cette variable. Autrement dit, la réservation de place en mémoire se fait avant le début de l'exécution du programme, ce qui peut poser un problème : imaginez que vous ayez par exemple besoin de deux tableaux ne pouvant tenir simultanément en mémoire ; vous n'en avez par ailleurs pas besoin simultanément, mais les déclarations se font en début de programme...

Aucune solution???

Vous allez me dire que ce qui serait bien, c'est de pouvoir allouer de la mémoire d'abord au premier tableau, puis libérer la place quand il ne sert plus afin de la réserver pour le deuxième tableau...

Hé bien tout ceci est possible grâce aux pointeurs... Cela porte le doux nom de...

1) Allocation dynamique de la mémoire

Bon, il est temps de commencer à parler un peu technique... Ce que j'ai décris plus haut est effectivement possible mais nécessite tout de même un peu de préparation...en effet, il faut "typer" le pointeur avant de pouvoir l'utiliser ; voici comment procéder :

(je donne l'exemple et j'explique ensuite !)

```
TYPE Mon_Type_de_donnée = INTEGER ;      (* par exemple ! *)
          Pointeur_sur_mon_type = ^Mon_type_de_donnée;

VAR ptr:Pointeur_sur_mon_type;
```

Ce que nous avons fait jusqu'à présent n'est que donner un type d'objet sur lequel le pointeur va pointer... des entiers en l'occurence... notez que Mon_type_de_donnée pourrait aussi bien être du type ARRAY ou encore RECORD (mais on verra ça plus loin...) Attention au "^" qui est très important... il indique qu'il s'agit d'un type sur lequel on va pointer et non un type que l'on va utiliser...

Ptr est donc un pointeur destiné à pointer sur un entier; Il est important de comprendre que ptr n'est pas un entier mais en quelque sorte une "flèche" qui va pointer sur une zone de la mémoire contenant un entier. Voyons comment on l'utilise...

```
          BEGIN
1              ptr:=NIL;
2              new(ptr);
3              writeln('donnez un entier:');
4              readln(ptr^);
5              writeln('vous avez donné :',ptr^);
6              dispose(ptr);
7              ptr:=NIL;
          END.
```

COMMENTAIRES :

La 1ère ligne est plus ou moins inutile mais une bonne habitude à prendre est d'initialiser ses pointeurs à NIL ; Les pointeurs non initialisés sont dits "pointeurs errants" et peuvent provoquer des plantages si on les utilise... NIL est concrètement une zone réservée de la mémoire qui ne contient rien.

La 2ème ligne crée une nouvelle variable dynamique à une adresse qu'elle se débrouille pour trouver puis elle affecte au pointeur ptr cette adresse afin que nous puissions utiliser la variable dynamique qui s'y trouve. C'est l'instruction NEW qui se charge de tout ça. Notez que c'est cette instruction qui est l' "instruction magique"... C'est elle comme vous le voyez qui permet de réserver de la mémoire alors que le programme est déjà lancé (d'où le nom d'Allocation Dynamique de la Mémoire...).

Pour utiliser une métaphore, on peut dire que NEW crée une boite de la taille d'un entier et indique sa position grâce au pointeur ptr. Notez qu'à partir du moment où le pointeur se voit affecter une adresse par NEW, il n'est plus considéré comme pointeur errant, d'où la quasi-inutilité de la ligne 1.

Nous apprenons dans les 4ème et 5ème lignes comment utiliser la variable fraîchement créée sur laquelle pointe ptr... Il suffit pour cela de la déréférencer en lui adjoignant le symbole ^.

Lorsqu'on déréférence un pointeur, ce n'est plus le pointeur que l' on considère, mais la variable sur laquelle il pointe. Pour reprendre la métaphore précédente, ptr indique la position de la boite et ptr^ désigne le contenu de la boite... Ce concept de déréférenciation est très important et est source de nombreuses erreurs pour les débutants (qui essayeraient par exemple d'affecter une valeur entière au pointeur au lieu de l'affecter à la variable dynamique désignée par le pointeur), alors relisez ce paragraphe jusqu'à l'avoir bien compris.

Si vous avez compris, vous voyez alors que la 4éme ligne affecte un entier à la variable dynamique pointée (syn. de désignée) par ptr, et que la 5ème ligne lit son contenu. Une variable dynamique se comporte exactement comme une variable classique si ce n'est sa syntaxe et sa façon d'y accéder un peu 'zarb'.

IMPORTANT : NE JAMAIS DEREFERENCER UN POINTEUR VALANT NIL (cela provoque immanquablement une erreur!)

La ligne 6 libère l'espace alloué en ligne 2 par NEW grâce à l'instruction DISPOSE. A partir de ce moment, la variable dynamique pointée par ptr n'existe plus... elle est tout simplement détruite. Ce qu'elle contenait est par conséquent perdu et ptr pointe à nouveau n'importe où...d'où l'utilité de le réaffecter à NIL en ligne 7.

Pour finir, quelques exemples de syntaxes selon les types des variables sur lesquelles un pointeur ptr pointe :

> Si ptr pointe sur une variable dynamique de type ARRAY[1..9], il faudra faire ptr^[15] pour accéder au 15ème élément du tableau sur lequel ptr pointe..
>
> Si ptr pointe sur une variable dynamique de type

```
Record
    x,y:integer;
end;
```

il faudra faire ptr^.x et ptr^.y pour accéder aux différents champs de la structure.

2) Mais où sont elles stockées, ces variables dynamiques?

Bonne question, je me remercie de l'avoir posée... Les variables dynamiques sont stockées sur le tas (HEAP en anglais) à ne pas confondre avec la pile (STACK en anglais) (La pile sert à gérer les appels et les retours des fonctions et procédures récursives). Allez d'ailleurs voir dans les options du Pascal (TP ou BP) la section memory ; vous constaterez que la place maximum allouée au tas est de 640Ko (tiens, ça ne vous rappelle rien?), ce qui laisse tout de même de la marge... mais ce n'est pas une raison pour la gaspiller, ce qui veut dire qu'il faut utiliser DISPOSE sur les variables dont on ne se sert pas le plus rapidement possible.

Je tiens à attirer votre attention sur la façon dont le tas est géré : supposons que vous remplissiez le tas avec des SHORTINT (codés sur 8 bits) puis que vous en supprimiez un sur deux... Essayez ensuite de créer une variable dynamique de type INTEGER... ça ne marchera pas pour cause de manque de mémoire... et pourtant, me direz-vous, 320Ko devraient suffire! Pourtant non... Le problème vient du fait que les données du tas doivent être disposées de façon continue et ne peuvent être fragmentées (donc hors de question de couper un integer de 16 bits en deux morceaux de 8 bits !). Ce qu'il faudrait faire est une sorte de défragmentation... Mon Pascal ne gère pas ça automatiquement mais peut-être que certains compilateurs le font... Je vous laisse le soin de faire une routine pour gérer ça... (il suffit en gros de reprendre toutes les variables, les détruire, puis les recréer...).

Je n'aborderai pas dans cet article les tests de mémoire disponible sur le tas... sachez seulement qu'il existe deux fonctions utiles, une qui donne la mémoire totale disponible (MemAvail), et une autre qui donne le plus grand bloc continu disponible (MaxAvail)... à utiliser dans les traitements d'erreurs avant d'utiliser NEW(pointeur), qui peut provoquer une erreur si la mémoire disponible est insuffisante.

3) Mise en garde contre les passages de paramètres

Comme vous le savez, il existe deux façons de passer des paramètres à une procédure ou fonction : Soit on les passe par adresse, soit on les passe par valeur. Par adresse (dans le cas où on utilise VAR avant le paramètre), le contenu de la variable peut être modifié par la fonction. Par valeur, une copie du paramètre est faite, et c'est sur cette copie que la fonction travaille, la valeur de l'original ne pouvant donc être modifiée.

Le passage de pointeurs comme paramètres ne déroge pas à cette règle mais les conséquences sont plus subtiles et peuvent causer des erreurs pénibles à détecter. Ceci est dû à l'erreur fréquente citée plus haut. Même si on passe un pointeur par valeur, c'est une copie du pointeur qui est faite, et non pas une copie de la variable dynamique sur laquelle il pointe ; ce qui provoque la chose suivante : il est tout à fait possible de modifier la valeur d'une variable dynamique dont le pointeur à été passé par valeur.

EX :

```
Program erreur_fréquente;

    TYPE : Mon_type_...
                Pointeur_sur... (* cf plus haut *)

    procedure
modifie_par_surprise(ptr:Pointeur_sur_mon_type);
    begin
        ptr^:=ptr^+1;
    end;

    var pointeur : Pointeur_sur_mon_type;

    begin
        new(pointeur);
        pointeur^:=5
        writeln('Valeur de la variable
avant :',pointeur^);
        modifie_par_surprise(pointeur);
        writeln('Valeur de la variable
après :',pointeur^);
        dispose(pointeur);
        pointeur:=NIL;
    end.
```

Le programme donnera comme valeur successivement 5 et 6... Comme quoi il faut plutôt être prudent...

Structures de Listes Chainées

Arthur Accroc

II) STRUCTURES DE LISTES CHAINEES

Je suppose que vous êtes en train de vous dire que ce que je raconte est bien amusant (quoique!), mais que nous ne sommes guère plus avancés, vu que pour l'instant, il faut déclarer autant de pointeurs que de variables dont on aura besoin... d'où un intérêt plutôt limité et une perte de temps évidente, même si c'est cool de réserver de la place soi-même... ce qui serait bien, ce serait de déclarer une seule variable de type pointeur dans le VAR et de pouvoir à partir de cette dernière en créer autant qu'on en veut... ça sent le récursif, tout ça, vous trouvez pas ?

Allez, c'est parti :

```
type TDonnee=integer;

type PCellule = ^Cellule;
          Cellule = record
                                  nb:TDonnee;
                                  next:Pcellule;
                                end;
```

Analysons un peu ce qui vient d'être écrit : un pointeur de type PCellule va pointer sur une structure contenant une donnée (du type que l'on veut) et un pointeur sur une autre Cellule (appellée également noeud dans la plupart des livres...) ; Cette structure permet la chose suivante : Le premier élément de la liste comportera un pointeur sur la deuxieme cellule, le deuxième comportera un pointeur sur la troisième cellule et ainsi de suite jusqu'au dernier élément dont le pointeur vaudra NIL, ce qui indiquera la fin de la chaîne de cellule.

Exemple concret ("fait à la main") :

Soit ptr une variable déclarée comme un pointeur de type PCellule.

```
new(ptr);      (* on    crée    une    variable    dynamique    structurée *)
ptr^.nb:=1; (* on affecte la partie donnée *)
```

ATTENTION,CA SE COMPLIQUE !!! (comme beaucoup de choses récursives !) Remarquez que ptr^.next est lui aussi un pointeur (errant pour le moment) du même type que ptr ; On peut donc lui assigner (si bien entendu il reste de la place sur le heap) une nouvelle variable dynamique ; ceci se fait naturellement de la façon suivante :

```
new(ptr^.next);
```

ptr^.next pointe donc à son tour sur un variable (dynamique) structurée ptr^.next^ semblable à ptr^ (c'est à dire à celle pointée par ptr).

On peut également lui affecter sa donnée :

```
ptr^.next^.nb:=2;
```

on peut continuer ainsi jusqu'à remplir le heap :

```
new(ptr^.next^.next);
```

Remarquez que la notation devient vite très lourde...

Pour se représenter une liste chaînée (c'est le nom qu'on leur donne), un bon moyen est de penser aux poupées russes qui s'emboitent les unes dans les autres, car, de même que pour obtenir la dernière poupée il faut toutes les ouvrir, pour accéder à la cellule n, il faut lire son adresse (=récupérer son pointeur) dans la cellule n-1...

Avant de passer aux algorithmes (récursifs, bien sûr !) permettant d'alléger la gestion de tout ça, je voudrais attirer votre attention sur la difficulté de gérer ces variables chaînées... Prenons l'exemple de la suppression d'une liste chaînée de deux éléments : il faut les supprimer en partant de la fin...

En effet : si vous détruisez ptr avec un dispose(ptr), vous perdez la trace de ptr^.next^ (car ptr^ n'existe plus !) et vous ne pouvez donc plus supprimer ptr^.next^, ne connaissant plus son adresse (qui se trouvait dans ptr^). La phrase précédente est un peu lourde, je m'en excuse ;-)

Ce qui est dit précédemment implique également le fait qu'IL NE FAUT JAMAIS PERDRE LE POINTEUR DE LA PREMIERE CELLULE.

D'ailleurs, dans les algos, nous travaillons toujours depuis la première cellule.

Dernier détail : le dernier élément d'une liste chaînée doit pointer sur NIL pour les deux raisons suivantes :

1) Sinon, c'est un pointeur errant (hihi, le prétexte !)

2) c'est un bon moyen d'indiquer la fin d'une liste. Cette convention
implique que la liste chaînée vide est NIL.

Voici le programme Pascal commenté contenant les fonctions usuelles pour traiter les listes chaînées... Si tout va bien, vous devriez le retrouver en Annexe (en version non commentée) avec les autres sources... (au pire, je vous laisse le soin d'en faire l'extraction à la main !)

Voir Listing 1, commenté..

Je vous conseille de tracer à la main sur du papier (en dessinant des boites avec leur contenu [les cellules] et des flèches [les pointeurs]) les trois dernières procédures pour bien comprendre leur fonctionnement récursif.

De toute façon, la mise en oeuvre est relativement simple, l'essentiel du code étant constitué de manipulations de pointeurs qui deviennent triviales dès que vous visualisez bien le principe de chaînage des noeuds... Par exemple, pour la suppression d'une cellule, il suffit de "sauter" le noeud à effacer et faire pointer le noeud précédent sur le noeud suivant... (faire en quelque sorte une coupure de chaque coté d'une cellule puis "recoller" les deux bouts de la chaîne...) Une fois ceci fait, le noeud désolidarisé de la liste peut être effacé !

Notez par ailleurs que les routines permettent de gérer plusieurs listes chaînées à la fois; il suffit pour cela d'utiliser plusieurs pointeurs.

N'oubliez pas que l'effacement du premier noeud ne détruit pas la structure... Au pire, si l'effacement est mal fait, vous perdez trace de la seconde cellule (et par conséquent des suivantes !), mais elles sont toujours en mémoire, même si elles sont devenues inaccessibles.

Une application possible des listes chaînées est le calcul en multi-précision (c'est à dire avec autant de chiffres qu'on le veut) en mettant un chiffre par cellule et en représentant chaque nombre par une liste chaînée...

Comme nous l'avons vu, le gros problème des listes chaînées est que si l'on perd le pointeur de la première cellule (en passant par ex. de façon maladroite à la suivante par une affectation du genre l:=l^.next), il devient impossible de la retrouver ; Il existe pour éviter ce genre de problème une structure qui permet de se promener librement sur la chaîne sans jamais en perdre un bout ; Cette structure, similaire à la précédente se nomme Liste Doublement Chaînée ; La principale différence se situe dans la structure d'une cellule : chaque cellule contient deux pointeurs :

- un qui pointe sur la cellule suivante
- un qui pointe sur la cellule précédente

Bien évidemment, la chaîne ainsi faite comporte un NIL à chaque extremité (le pointeur 'previous' de la première cellule et le pointeur 'next' de la dernière pointe les deux sur NIL). L'avantage de cette structure est qu'elle donne plus de liberté, qu'elle est plus rapide à utiliser, mais elle est en contrepartie plus lourde à gérer et elle prend plus de place en mémoire (deux pointeurs dans chaque cellule au lieu d'un seul... ce qui peut paraître excessif dans le cas de cellules contenant des entiers mais qui devient tout à fait raisonnable dans le cas des structures "encombrantes").

Voici un programme qui permettrait de gérer des listes doublement chaînées... Vous le retrouverez également en annexe... Les commentaires seront ici plus rares

Voir Listing 2, commenté..

Voilà ! C'est tout pour les listes chaînées... Notez que les deux petits programmes fournis sont parfaitement incomplets (surtout le deuxième qu'il faut plus prendre comme quelque chose destiné à donner une idée des complications qu'entraine le double chaînage...). Il manque en particulier des tests concernant la place disponible sur le tas. Vous pourrez vous-même programmer des fonctions de recherche et de tri (le travail n'est pas mal maché dans le cas d'un tri par insertion, je crois...). En gros, vous avez maintenant les bases, tout reste à faire (à moins qu'un gentil ROR se dévoue dans un prochain article...).

Il ne nous reste plus à voir pour aujourd'hui les structures de données classiques que sont les piles (encore appelées LIFO) et les files d'attentes (appelées quant à elles LILO)

Piles et files d'attente

Arthur Accroc

III) PILES ET FILES D'ATTENTES

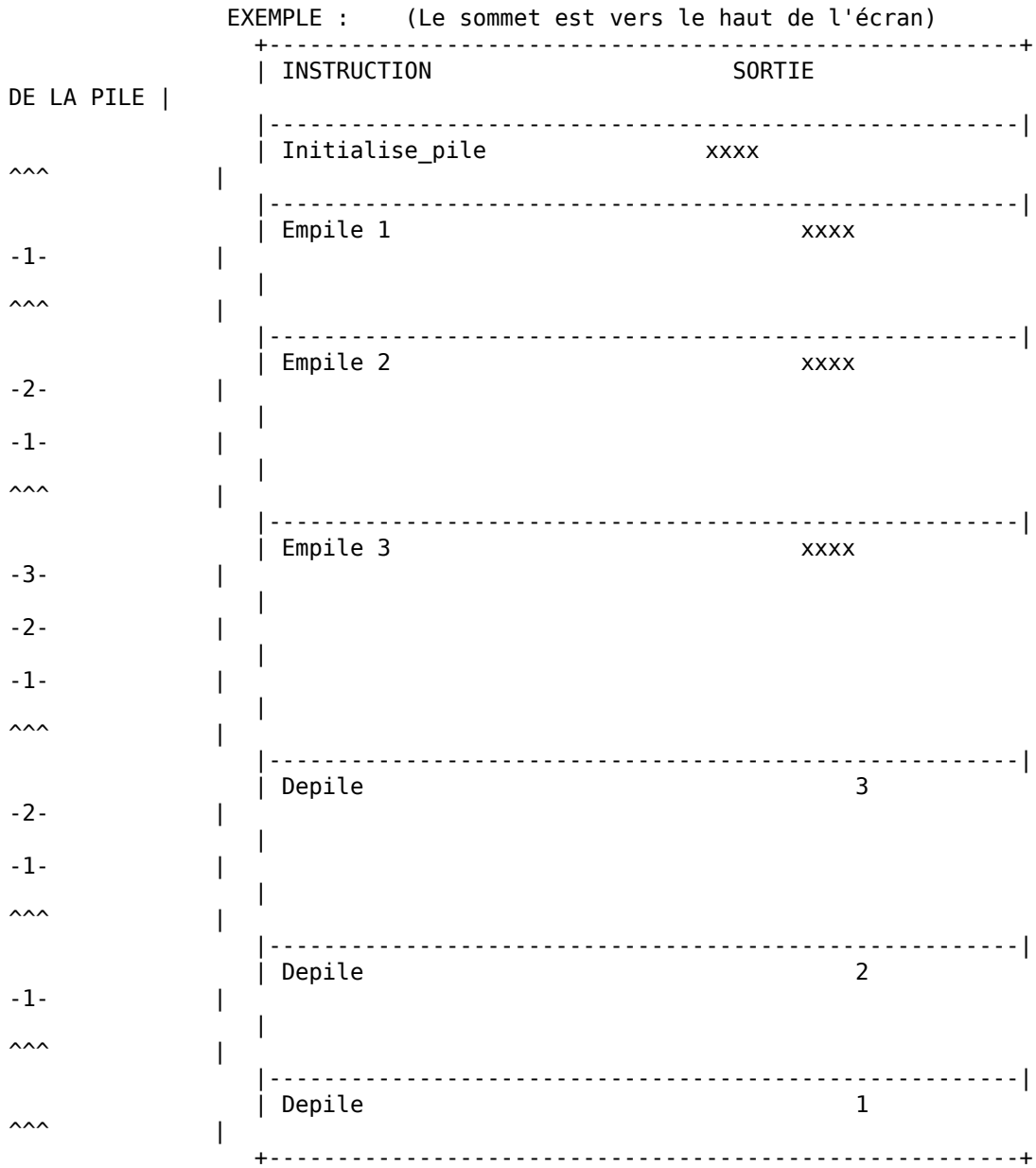
Contrairement à ce que vous imaginez, nous n'allons pas parler de volts ou d'attentes interminables aux caisses des supermarchés ;-) C'est pas de ma faute si ces structures s'appelent comme ça ! En fait, pour vous donner une petite idée préalable, disons que les piles tiennent plus de l'empilement que des batteries, et que les files d'attentes ont quand même quelque chose à voir avec les queues des supermarchés dans le sens où elles illustrent la philosophie du "Premier arrivé, premier servi".

Commençons par le plus simple (à mettre en oeuvre 8->)

1) Les piles (LIFO)

Les structures de pile ne devraient pas sembler étrangères aux utilisateurs de calculateurs HP... c'est une structure dans laquelle on empile et dépile des données. L'endroit par lequel entrent et sortent les données est appelé le Sommet. On appelle profondeur de la pile le nombre d'éléments qu'elle contient. Comme les opérations (empilage et dépilage) se font toutes au niveau du sommet, cette structure est aussi appelée LIFO - Last In First Out (Dernier rentré Premier sorti) -. Une bonne représentation d'une pile est la pile de plateaux dans les caféterias... pour accéder au deuxième plateau, il faut d'abord retirer le premier.

Une propriété essentielle des piles est qu'elles restituent les données rentrées à l'envers :



Nous avons donc empilé 1-2-3 et la pile nous a ressorti 3-2-1, donc elle a rendu la séquence de nombres à l'envers.

Passons maintenant à l'implémentation...

Pour les piles (et les files), j'ai préparé une unité prête à être utilisée; mais ce n'est pas parce que j'ai tout fait qu'il n'y a plus rien à faire... Comme précédemment, je vous laisse le soin d'améliorer le code en y adjoignant les tests pour vérifier si la mémoire du tas est suffisante. J'ai volontairement omis d'implanter une fonction qui retourne la profondeur (très facile à faire !) et vous devrez surement faire quelques modifications mineures si vous souhaitez utiliser autre chose que des entiers comme données..

Cependant, l'unité telle qu'elle est fonctionne parfaitement.

Vous trouverz une version non commentée en annexe. Voici la version commentée:

Notez que nous n'avons besoin à chaque fois de ne passer que le pointeur qui pointe sur le sommet de la pile.

Voir Listing 3, commenté..

Voila, comme vous le voyez, c'est très court ! Je vous propose à titre d'exercice de vous amuser à implanter les procedures suivantes:

- SWAP : interverti le sommet et la deuxième case de la pile

- ROT : fait une permutation circulaire sur les 3 premiers élements en partant du sommet.

- DEEP : retourne la profondeur de la pile

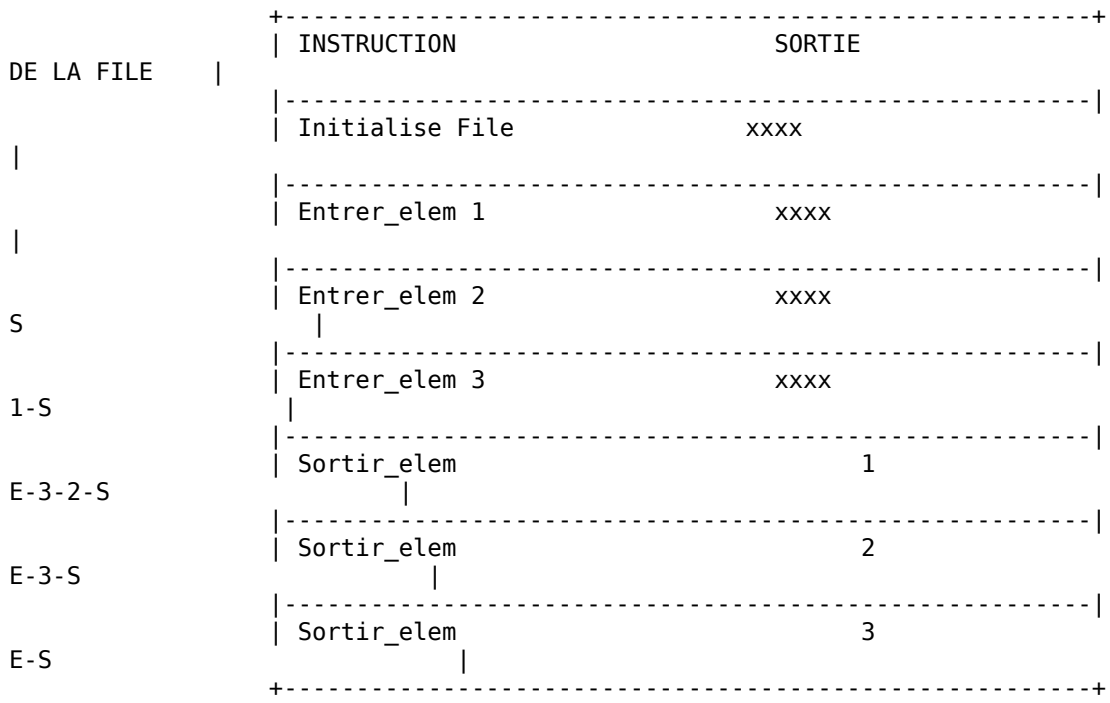
Le champ d'application des piles est vaste... Elle peuvent servir par exemple à stocker des résultats intermédiaires ou encore être utilisées pour faire des analyseurs syntaxiques...

"And now, something different..." (J. Clees)

2) Les files d'attente

Le principe des files d'attente (ou queues) est simple ; pour continuer avec mes comparaisons, une file d'attente est comme une pile, mais avec une entrée et une sortie... C'est à dire que les données s'accumulent dans la file d'attente jusqu'à ce qu'elle soient traitées (sorties). Les premières données arrivées sont donc les premières traitées, les autres s'accumulant derrière et attendant leur tour... C'est pour cela que cette structure est également appelée FIFO - First In First Out (Premier rentré Premier sorti).

Je vais refaire le même shéma que pour les piles mais en utilisant cette fois-ci une file. La représentation est horizontale. E indique l'entrée et S la sortie...



Vous remarquerez que dans le cas de la structure de file d'attente, les données ressortent dans l'ordre dans lequel elles sont rentrées (contrairement aux piles).

Passons à l'implémentation :

J'ai à nouveau fait une unité minimale que vous retrouverez prête à l'emploi dans les annexes...

Je vous demanderai de bien noter que l'implantation de cette structure est plus délicate que celle de la pile donc faites plus attention ! Cette fois ci, nous avons besoin de deux pointeurs :

- un qui pointe sur la tête de la liste chaînée (head)
- un qui pointe sur la fin de la liste chaînée (tail)

Voir Listing 4, commenté..

Et c'est tout pour les structures de files...

Ces structures ont également beaucoup d'applications ; elles peuvent par exemple servir à stocker des données en attente de traitement (si le processeur est trop occupé pour les traiter pour l'instant...) ou peuvent également servir de stockage temporaire... A vous de trouver les meilleures applications ou plutôt, essayez de voir où elles peuvent servir maintenant que vous connaissez leur existance...

Conclusions

Arthur Accroc

IV) CONCLUSIONS

Voila, c'est tout pour cette fois... Je tiens tout de même à faire une ou deux remarques concernant la programmation utilisant les pointeurs...

Comme nous l'avons vu, les pointeurs permettent une utilisation plutôt pratique de la mémoire mais il y a des 'mais'...

1. Même si une structure de liste chaînée (par exemple) peut sembler plus pratique qu'un tableau (en particulier dans les cas d'insertion d'éléments), l'accès à la liste chaînée est globalement beaucoup plus lent que l'accès à un élément de tableau ; ça peut parfois avoir son importance...

2. La mise en oeuvre des pointeurs est relativement pénible car les bugs sont généralement assez vicieux ; je m'explique : Déréférencer un 'pointeur errant' ne provoque pas d'erreur à la compilation mais peut provoquer lors de l'exécution a peu près n'importe quoi depuis "no problemo" jusqu'à "même Ctrl-Alt-Suppr ne fonctionne plus"... ce qui peut conduire à des méchantes crises de nerfs quand un bug oublié oblige à re-booter pour la 15ème fois son PC (qui, rappelons-le, fait tout pour retarder au maximum le moment où on le fera travailler à grands renforts de 'memory test' et autres 'floppy seek'...). En résumé, les programmes utilisant les pointeurs sont globalement plus pénibles à débugger que les autres...

N'oubliez pas qu'il vaut mieux passer du temps à écrire ses algos sur du papier (en pseudo-code, comme dirait Legolas ;-) que directement sur la machine. Cette remarque générale s'applique à plus forte raison ici ; n'oubliez pas que perdre du temps sur le papier, c'est en gagner par la suite...

D'autre part, vous n'avez découvert dans cet article que la partie visible de l'iceberg... Le champ d'application des pointeurs est énorme (allez d'ailleurs faire un tour du coté des programmeurs en langage C... eux, les pointeurs, ils connaissent (ils n'ont d'ailleurs pas tellement le choix '-( ) (oulala, faut que je me calme avec les parenthèses, ça va finir par ressembler à du LISP;-)))

NOTE : En parlant du LISP, si vous comptez profiter de la série d'articles d'initiation au Scheme pour vous y mettre, la compréhension des listes chaînées vous permettra de moins 'nager' dans les listes du Scheme ... (pff... tant de disgressions pour en arriver là...!)

Au menu pour la prochaine fois : les arbres (si j'ai trouvé d'ici là : a) le temps de couper le grand poil qui me gêne à chaque fois que je tape sur le clavier
b) un Gentil ROR qui puisse m'aider parce que ça risque d'être un gros article vu ce qu'il y a à raconter sur le sujet)

A+ et d'ici là, pointez bien !

-=≡ Arthur Accroc ≡=-

```
program list_s_c; (* liste simplement chaînée *)

uses crt;

type TDonnee=integer;

type PCellule = ^Cellule;
          Cellule = record
                                    nb:TDonnee;
                                    next:Pcellule;
                                end;

function init_liste:PCellule;
(* initialise une liste *)
begin
    init_liste:=nil;
end;

function est_vide(l:Pcellule):boolean;
(* retourne true si la liste est vide *)
begin
    est_vide:= (l=nil);
end;

function suivant(l:Pcellule):Pcellule;
(* retourne l'adresse de la cellule suivante *)
begin
    suivant:=l^.next;
end;

function contenu(l:Pcellule):TDonnee;
(* retourne le contenu d'une cellule pointée par l *)
begin
    contenu:=l^.nb;
end;

(* bon... ça devient, sérieux, va falloir commenter un peu plus!! *)

function insere(l:Pcellule;n:TDonnee):PCellule;
(* permet d'insérer une cellule contenant une donnée n devant la cellule sur
laquelle pointe l ; Est retourné le pointeur de la nouvelle liste *)
var temp:PCellule;
begin
    new(temp);      (* on crée une nouvelle variable dynamique *)
    temp^.nb:=n; (* on lui affecte son contenu *)
    temp^.next:=l; (* on la fait pointer sur la 1ère cellule qui devient la 2ème *)
    insere:=temp;    (* cette cellule devient ainsi la 1ère (c'est son pointeur que
                                    l'on retourne comme tête de liste) *)
end;

function supprime(l:Pcellule):Pcellule;
(* permet de supprimer la cellule pointée par l ; Est retourné le pointeur de
la nouvelle liste *)
var temp:Pcellule;
begin
    temp:=l^.next;    (* on garde D'ABORD trace de la cellule suivante (elle va
                                      perdue à la destruction de l^ !) *)
```

```
    dispose(l);          (* on détruit la variable pointée par l *)
    supprime:=temp; (* la cellule suivant l (et préalablement stockée dans temp)
                                    devient la nouvelle tête de liste *)
end;

(* c'est maintenant que l'on saute à pied joints dans le récursif !!
accrochez vous à vos synapses, les neuronnes vont chauffer ! *)

function supprime_elem(l:PCellule;elem:TDonnee):PCellule;
(* fonction qui parcours la liste ; si elle trouve une cellule contenant la
donnée elem, elle la supprime ; elle retourne un pointeur pointant sur la
première cellule de la nouvelle liste *)
begin
if est_vide(l)              (* liste vide? *)
    then supprime_elem:=l        (* on change rien ! *)
    else if contenu(l)=elem    (* SINON *) (* si la première cellule est la bonne *)
          (* (remarque : /\ attention au test valable seulement pour type scalaire)
*)
                then supprime_elem:=supprime(l) (* alors on supprime la cellule *)
                  else begin                                    (* sinon *)
                            l^.next:=supprime_elem(suivant(l),elem); (* on recommence
mais en

partant de la cellule suivante ...
                                                          notez que c'est là
qu'intervient la récursivité !
                                                          on fait pointer la cellule
courante sur la cellule
                                                          qui fera partie de la liste
traitée (dont le
                                                          pointeur est retourné par
supprim_elem !) ; en
                                                          gros, on recolle les bouts! *)
                            supprime_elem:=l;        (* et on retourne la liste
traitée *)
                            end;
end;

function insere_sort(l:PCellule;elem:TDonnee):PCellule;
(* insère une cellule contenant elem à la bonne place et retourne le pointeur
pointant sur le début de la nouvelle liste *)
begin
    if est_vide(l)        (* si la liste est vide (on si on est auu bout de la liste)
*)
        then insere_sort:=insere(l,elem)    (* on crée une liste contenant l'élément
*)
        else if elem<=contenu(l)    (* sinon si l'élément est plus petit que le

contenu de la cellule courante *)
        (* REM : RELATION D'ORDRE A DEFINIR SELON LE TYPE DE TDonnee !!! *)
                    then insere_sort:=insere(l,elem) (* on insère une cellule
contenant

l'élément et on retourne son

pointeur *)
                    else begin                                                (*
SINON *)
                                l^.next:=insere_sort(suivant(l),elem); (* même
remarque que
```

```
pour la fonction précédente !
                                                                        C'est
là qu'intervient la récursivité
                                                        On fait pointer la
cellule courante sur la
                                                        cellule qui fera
partie de la liste traitée *)
                                insere_sort:=l; (* on retourne la liste traîtée *)
                                end;
end;

procedure affiche(l:Pcellule);
(* procédure affichant le contenu de toutes les cellules depuis celle pointée
par l jusqu'à la fin de la liste *)
begin
if not(est_vide(l))                 (* si la liste n'est pas vide (équivalent à
savoir
                                                        si on a atteint la
fin qui est NIL selon la
                                                        convention adoptée *)
then begin
          writeln(contenu(l));      (* on affiche le contenu de la cellule courante
*)
          affiche(suivant(l));(* on ré-appelle la procédure pour la cellule suivante
*)
          end;
end;

(* ce qui suit est un démonstration de l'utilisation
des procédures précédentes *)

var l:pcellule;

BEGIN
clrscr;
l:=init_liste;    (* très important car sinon le dernier élément
                                ne pointe pas sur NIL ! *)
l:=insere(l,5);
l:=insere(l,4);
l:=insere(l,3);
affiche(l);
l:=supprime_elem(l,4);
writeln;
affiche(l);
l:=insere_sort(l,9);
writeln;
affiche(l);
END.
```

```
program list_d_c;    (* listes doublement chaînées *)

uses crt;

type TDonnee=INTEGER;

type PCellule = ^Cellule;
          Cellule = record
                                nb:integer;
                                prec,suiv:PCellule;
                             end;

function init_liste:PCellule;
(* fonction qui retourne une liste vide *)
begin
init_liste:=nil;
end;

function insere(l:PCellule;elem:TDonnee):PCellule;
(* fonction insérant un élément elem devant la cellule pointée par l et
retourne le pointeur de la nouvelle liste ainsi crée *)
var temp:PCellule;
begin
    new(temp);                  (* création nouvelle cellule *)
    temp^.nb:=elem;                (* affectation *)
    temp^.suiv:=l;            (* recollement avec la suite *)
    if l<>nil              (*    si liste vide    *)
        then    begin
                    temp^.prec:=l^.prec;          (* on crée un liste... *)
                    l^.prec:=temp;                     (* ...à un élément *)
                    end
        else temp^.prec:=nil;          (* s'il y a des cellules derriere *)
    if temp^.prec<>nil                     (* on recolle alors aussi ... *)
        then temp^.prec^.suiv:=temp; (* ...de l'autre coté *)
    insere:=temp;
end;

function supp_prec(l:PCellule):PCellule;
(* fonction supprimant la cellule précédant celle pointée par l *)
(* Est retourné un pointeur pointant sur la nouvelle liste *)
var ptr,ptr_prec:PCellule;
begin
    if (l=nil) or (l^.prec=nil)        (* si pas de précédente ou liste vide *)
        then supp_prec:=l                           (* on change rien *)
        else begin                                       (* sinon *)
                  ptr:=l^.prec;                        (* on garde trace... *)
                  ptr_prec:=ptr^.prec;          (* ...de ce qui précède... *)
                  ptr_prec^.suiv:=l;              (* ...de ce qui suit... *)
                  l^.prec:=ptr_prec;              (* ...on recolle... *)
                  dispose(ptr);                        (* ...puis enfin on fait du
ménage! *)
                  supp_prec:=l;
                  end;
end;

function supp_suiv(l:PCellule):PCellule;
(* pareil que la précédente mais de l'autre coté (donc no comment!) *)
var ptr,ptr_suiv:PCellule;
```

```
begin
    if (l=nil) or (l^.suiv=nil)
        then supp_suiv:=l
        else begin
                  ptr:=l^.suiv;
                  ptr_suiv:=ptr^.suiv;
                  l^.suiv:=ptr_suiv;
                  ptr_suiv^.prec:=l;
                  dispose(ptr);
                  supp_suiv:=l;
                  end;
end;

function debut(l:PCellule):PCellule;
(* fonction retrouvant le pointeur de la première cellule *)
(* notez que c'est exeptionellement de l'itératif ! *)
begin
    if l<>nil then while (l^.prec<>nil) do l:=l^.prec; (* dur à comprendre ? *)
    debut:=l;
end;



procedure aff(l:PCellule);
(* procédure d'affichage de tous les éléments... également itératif pour
changer un peu ! *)
begin
    l:=debut(l);
    while l<>nil do
        begin
            writeln(l^.nb);
            l:=l^.suiv;
        end;
end;

(* ce qui suit montre comment utiliser les fonctions précédentes... *)

var l,p:PCellule;

BEGIN
clrscr;
l:=nil;
l:=insere(l,1);
l:=insere(l,2);
l:=insere(l,5);
l:=insere(l,6);
aff(l);
writeln;
l:=supp_prec(l^.suiv);
aff(l);
writeln;
l:=supp_suiv(l);
aff(l);
END.
```

```
unit LIFO;      (* unit de gestion de pile *)

INTERFACE

type TDonnee = integer;

type Pile = ^Tcase;
          Tcase = record
                          n:TDonnee;
                          next:Pile;
                          end;

procedure init_pile(var p:pile);      (* initialisation de la pile *)
                                                                    (* attention,
l'initialisation ne vide
                                                                    pas la pile,
elle remet juste le
                                                                    pointeur au
depart...
                                                                    Pour vider la
pile, utiliser plusieurs
                                                                    fois la
procedure depile ! *)

function pile_vide(p:pile):boolean; (* retourne true si la pile est vide *)

procedure empile(var p:pile;nb:TDonnee);
(* empile un élément nb au sommet de la pile p *)

procedure depile(var p:pile; var nb:TDonnee;flag:boolean);
(* depile un element de la pile p. Flag retourne true si l'operation a
réussi (si la pile n'était pas vide) *)

IMPLEMENTATION

procedure init_pile(var p:pile);
begin
    p:=nil;              (* c'est encore une fois une convention... *)
end;

function pile_vide(p:pile):boolean;
begin
    pile_vide:=(p=nil);    (* découle de la convention précédemment citée... *)
end;

procedure empile(var p:pile;nb:TDonnee);
var pile_aux:pile;
begin
    new(pile_aux); (* on crée une nouvelle case *)
    pile_aux^.n:=nb;    (* on lui affecte son contenu *)
    pile_aux^.next:=p; (* on la fait pointer sur ce qui était la première case *)
                                                    (* et qui descent ainsi en profondeur
2 *)
    p:=pile_aux;    (* on fait de la nouvelle case le sommet de la pile *)
end;
```

```
procedure depile(var p:pile; var nb:TDonnee;flag:boolean);
var pile_aux:pile;
begin
    if pile_vide(p)
        then flag:=false            (* si pile vide, on ne valide pas le flag *)
        else begin                    (* SINON *)
                  flag:=true;              (* on valide le flag *)
                  pile_aux:=p;            (* on fait pointer pile_aux sur le sommet
*)
                  nb:=pile_aux^.n;    (* on extrait la donnée dans nb *)
                  p:=pile_aux^.next;(* on fait pointer p sur le deuxieme element *)
                  dispose(pile_aux);(* on supprime le premier element *)
                  end;
end;

END.
```

```
unit FIFO; (* files d'attentes ou queues *)

INTERFACE

type TDonnee = Integer;

type PQueue = ^Cellule;
          Cellule = record
                                  nb:TDonnee;
                                  next:PQueue;
                              end;

procedure init_queue(var head,tail:PQueue); (* initialise la queue *)

procedure entrer_elem(var head:PQueue;var tail:PQueue;item:TDonnee);
(* entre un élément 'item' dans la file d'attente *)

procedure sortir_elem(var head:PQueue;var tail:PQueue;var item:TDonnee;var
flag:boolean);
(* ressort le prochain élément de la fine d'attente dans la variable item.
Le flag est true si l'opération c'est bien déroulée (autrement dit, s'il y
avait encore du monde dans la file ;-) *)

IMPLEMENTATION

procedure init_queue(var head,tail:PQueue);
begin
    head:=nil;                  (* conventions conventions... *)
    tail:=nil;
end;

procedure entrer_elem(var head:PQueue;var tail:PQueue;item:TDonnee);
var ptr:PQueue;
begin
    new(ptr);                                           (* création... *)
    ptr^.nb:=item;                                   (* ...et remplissage... *)
    ptr^.next:=nil;                                 (* ...de la nouvelle cellule *)
    if head=nil                              (* si c'est le premier de la file... *)
        then head:=ptr
        else tail^.next:=ptr;
    tail:=ptr;
end;

procedure sortir_elem(var head:PQueue;var tail:PQueue;var item:TDonnee;var
flag:boolean);
var ptr:PQueue;
begin
    if head=nil                                    (* si la file est vide *)
        then flag:=false           (* on signale qu'on ne peut rien sortir *)
        else begin                      (* SINON *)
                      flag:=true;        (* on signale que ça marche *)
                      ptr:=head;                        (* on découpe... *)
                      head:=head^.next;              (* ...on recolle *)
                      item:=ptr^.nb;               (* on sort le contenu dans une
variable *)
                      dispose(ptr);                   (* et on fait du ménage... *)
                 end;
```

```
end;

END.
```

Initiation au Scheme (I)

Arthur Accroc

Initiation au langage Scheme

schème : [...] - (Philosophie) Représentation intermédiaire entre le concept et les données de la perception.

(Petit Robert)

Préambule...

Init.au Scheme, Préambule...

Arthur ACCROC

J'espère par cette série d'articles vous faire découvrir un langage qui a totalement changé ma vision de la programmation. Le Scheme est un langage fabuleux pour maîtriser la récursivité. D'autres part, il permet de faire des choses impensables dans d'autres langages tels que le BASIC, le C ou le PASCAL... Evidemment, n'imaginez pas programmer une mégademo ou un clone de DOOM, ce n'est pas le but de ce langage... Par contre, je pense que les amateurs de programmation élégante (voire même 'tordue') en apprécierons les finesses... Si à un moment ou à un autre de la lecture de cet article vous vous sentez perdus, pas de panique ! Certains concepts abordés seront détaillés plus loin... Si vous ne comprenez toujours pas, adressez vous à moi ou à tout autre 'Schemeur' !
Commençons par un petit peu de culture générale

Abréviation de LISt Programming ou Lot of Insane and Stupid Parentheses selon l'humeur...

Au sens mathématique : Ensemble exhaustif de toutes les clauses ou termes propres à définir ou à intervernir dans la résolution d'un pro- blème.

Exemple de tautologie :

```
      VAR
    X: Integer;
    BEGIN
      IF X<0 THEN
          ....
      ELSIF X=0 THEN
          ....
      ELSIF X>0 THEN
          ....
      END;
    END;
```

Cette série de tests est une tautologie car elle est en mesure de déclencher une action quelque soit la valeur de X :

X<0; X=0 ; X>0

Au sens littéraire :

Répétition inutile d'une même idée en termes différents.

Dictionnaire Usuel
Flammarion

Init.au Scheme, culture générale

Arthur Accroc

Le SCHEME (prononciation : [skim]) est un langage de la famille des langages LISP dont la toute première version fut développée dès la fin des années 50 au Massachusetts Institute of Technology (au M. I. T.) par John McCarthy et ses associés, ce qui fait du LISP un des deux plus anciens langages encore utilisés (l'autre étant le FORTRAN).

Les langages LISP se distinguent essentiellement par leur capacité à manipuler aussi bien des symboles que des nombres, ce qui les a rendus à l'époque très populaires dans le domaine de l'I.A. (Intelligence Artificielle). Bien entendu, maintenant, on préfère les langages comme le PROLOG pour de telles applications. Nous pouvons en fait dire que les langages LISP se situent quelque part entre la programmation procédurale (Pascal, C, etc..) et les langages de PROgrammation LOGique.

En LISP, une des particularités (qui saute aux yeux !) est l'utilisation très importante de la structure de données nommée liste.

Les listes sont très importantes en LISP non seulement parce qu'elles sont une façon très pratique et flexible de manipuler des données, mais aussi pour une raison 'subtile' dont la puissance deviendra apparente plus tard dans cette initiation : En LISP, les programmes sont eux mêmes représentés par des listes, ceci ayant pour conséquence que les programmes LISP peuvent être traités comme des objets par d'autres programmes LISP.

Quant aux différences entre le SCHEME et les autres 'dialectes' LISP, elles sont minimes et il est inutile de les détailler ici car elles n'ont aucune importance dans le cadre d'une initiation (elles sont pour la plupart syntaxiques). Le Scheme est plus récent que le Lisp commun, c'est pour cela que je l'ai choisi plutôt que le Lisp...

Les exemples qui seront donnés tout au long de cette initiation prendront comme base l'implémentation PC-Scheme v.3 de ce langage. Pour d'autres implémentations (telle que celle fournie avec Linux), il ne devrait pas y avoir de problèmes majeurs d'adaptation.

Dernière remarque : cette implémentation du scheme n'est pas case- sensitive (c'est à dire qu'elle ne différencie pas les majuscules des minuscules comme le ferait le C), et j'ai donc pris quelques libertés syntaxiques à ce niveau. Je ne peux cependant pas garantir qu'il en soit de même pour toutes les implantations ! (méfiez-vous de celles sous UNIX car sous cet OS, presque tout est case-sensitive!!)

Ce premier volet de l'initiation va se dérouler de la façon suivante :

Init. au Scheme (I), Prise en main

Arthur Accroc

1) PRISE EN MAIN :

PC-Scheme, une fois lancé, affiche quelque chose qui doit ressembler à ceci :

```
PC Scheme 3.0
(C) Copyright 1987 by Texas Instrument
        All Right Reserved.

[PCS-DEBUG-MODE is OFF]
[1]
```

PCS (PC-SCHEME sera dorénavant abrégé de cette façon) attend des arguments. Le [1] est un compteur et va s'incrémenter à chaque opération. Si l'on tape 3, PCS retourne 3 et attend de nouveau :

```
[1] 3
3
[2]
```

PCS a en fait évalué ce que l'utilisateur a tapé. On peut ainsi se servir de PCS comme d'une calculatrice, sachant qu'il faut utiliser une notation algébrique préfixée ; voici quelques exemples (ne vous attardez pas sur les détails pour l'instant, tout sera clarifié au chapitre suivant !!!)

```
                        [1] (+ 1 2)
                        3                                   <- en effet l'évaluation
de (+ 2 1)
                        [2]                                     retourne 3 car 2+1=3 !
```

essayons ensuite :

```
                        [2] (+ 1 (* 4 5))
                        21                                  <-- Idem...
                        [3]
```

Si l'on tape A, PCS retourne ceci :

```
                        [1] a

                        [VW ERROR encountered !] Variable not défined in
                        lexical environment
                        A
                        [Inspect]
```

La frappe de A a provoqué une erreur car A n'est pas encore défini. Une erreur s'est produite, PCS entre donc automatiquement en mode Inspect (qui sera peut-être détaillé plus tard). Pour quitter ce mode, faire [CTRL]+Q (les curieux peuvent taper '?'). On revient ainsi à l'invite de l'interpréteur.

Nous allons maintenant définir A.

```
                        [1] (défine a 4)
                        A
                        [2]
```

Nous avons défini la variable A ou pour être plus exact, nous avons créé un lien entre le symbole a et la valeur 4.

Si l'on tape a, PCS renvoie 4, de même que (+ a 1) retourne 5 (Vous pouvez remarquer que je suis en train de choper la flemme et que j'arrête de tout écrire en détail ;-) Je vais d'ailleurs adopter la notation suivante : dorénavant, j'abrège "retourne" en "==>".

Juste encore quelques mots sur PCS, mais pas des moindres ! Nous serons amenés plus tard à écrire des programmes plus importants. Dans l'exemple précédent, par exemple, si nous avions tapé (define a 5) plutôt que (define a 4), pour corriger, il aurait fallu retaper entièrement (define a 4) (comme je suis en train de le faire en ce moment ;-). Il existe une solution plus simple : PCS contient un éditeur de texte intégré nommé edwin.

Pour le lancer, il suffit de taper (edwin) ; on arrive alors dans l'éditeur edwin qui est un éditeur de texte très classique quoique peu convivial de premier abord.

Voici ces principales commandes :

```
[ESC] O            : Sortie de edwin avec évaluation de son contenu.
[CTRL]+Pause : Sortie de edwin sans évaluation de son contenu.
[CTRL]+X+V      : Ouvre un fichier ASCII.
[CTRL]+X+W      : Sauve dans un fichier ASCII le contenu de edwin.
[CTRL]+X+S      : Idem mais sans demander le nom du fichier destination
                              (prend par défaut le dernier nom utilisé).
```

Ce sont a peu près les seules commandes indispensables de edwin.

Nous avons terminé ce petit tour de PCS. Pour quitter PCS, taper la commande (exit).

Init. au Scheme, Un peu de syntaxe

Arthur Accroc

II) Un peu de syntaxe :

Nous allons ici présenter ce qui sert de base à toute la syntaxe du SCHEME (et des LISP en général...). Les utilisateurs de calculateurs HP ne devraient pas trop être dépaysés. Le but de la lambda-notation est de fournir une manière formelle et unique de décrire une fonction.

1) Forme algébrique préfixée :

La fonction doit être une forme algébrique préfixée ; c'est à dire que l'opérateur doit précéder les opérandes. Voici quelques exemples :

```
1+2          s'écrit (+ 1 2)
1/2          s'écrit (/ 1 2)
3/(5+7) s'écrit (/ 3 (+ 5 7))
```

On peut remarquer que le nombre de parenthèses devient rapidement énorme. Ca aussi, c'est une des caractéristiques importante des LISP ! Alors un conseil : faites gaffe à vos parenthèses : à une parenthèse ouverte doit toujours correspondre une et une seule parenthèse fermée.

Voici ici le détail des principales fonctions mathématiques

2) lambda notation :

La notation lambda a la forme suivante :

```
(lambda (var1 ... varn) fonction_en_notation_préfixée)
```

Voyons quelques exemples :

(lambda (x) (* x x)) est la fonction qui à x associe x^2. Si l'on évalue cette fonction, on obtient comme résultat d'évaluation #<PROCEDURE>. En effet, les formes lambda sont des procédures (synonyme de fonctions) car elles peuvent être appliquées à un argument. Pour appliquer une fonction à un argument, nous avons vu qu'il fallait utiliser la notation préfixée. Donc, pour appliquer x^2 au nombre 5, il faut taper :

```
((lambda (x) (* x x)) 5)
```

ce qui retournera 25, (lambda [...]) ayant pris 5 comme argument.

Nous pouvons par ailleurs nous amuser à définir une fonction plus (+) prenant seulement deux arguments (contrairement à la fonction primitive plus qui prend n arguments) de la manière suivante :

```
(lambda (a b) (+ a b))
```

(on remarquera ici la profonde inutilité de cette fonction...)

Initiation au Scheme (I), principales fonctions

Arthur Accroc

(TRES) PETIT FORMULAIRE

*

format: (*)
        (* r1 r2 ... rn)

        Lorsque la fonction "*" est appelée sans argument, elle retourne
        l'élément neutre de la multiplication, c'est à dire 1. Avec n
        arguments, elle retourne r1 * r2 * ...   * rn.

+

format: (+)
        (+ r1 r2 ... rn)

        Lorsque la fonction "+" est appelée sans argument, elle retourne
        l'élément neutre de l'addition c'est à dire 0. Avec n arguments, elle
        retourne r1 + r2 + ... + rn.
        (et vive le copier-coller !)

-

format: (- r1 r2 ... rn)

        Lorsque la fonction "-" est appelée avec un seul argument, elle
        retourne l'opposé de cet argument. Avec n arguments, elle retourne
        r1 - (r2 + ...   + rn).

                Ex : (- 1)   ==> -1
                          (- 3 2)   ==> 1
                          (- 2 3 1) ==> -2 car 2-(3+1)=-2

/

format: (/ r1 r2 ... rn)

        Lorsque la fonction "/" est appelée avec un seul argument, elle
        retourne l'inverse de cet argument. Avec n arguments, elle retourne
        r1 / (r2 * ...   * rn).

                Ex : (/ 2)   ==> .5
                          (/ 3 2)   ==> 1.5
                          (/ 4 2 2) ==> 1 car 4/(2*2)=1

QUOTIENT

format: (quotient nb1 nb2)

        Si nb2<>0, retourne le quotient entier de nb1 par nb2, ce quotient
        ayant le signe du produit de nb1 par nb2.

                Ex : (quotient 12 3) ==> 4
                          (quotient 22 7) ==> 3

REMAINDER

format: (remainder nb1 nb2)

            Si nb1<>0, retourne le reste de la division de nb1 par nb2, ce reste
            ayant le signe de nb1.

```
            Si (quotient nb1 nb2) ==> q
            et (remainder nb1 nb2) ==> r
            alors (+ (* q nb2) r) ==> nb1.
```

MODULO
format: (modulo nb1 nb2)

Si nb2<>0, retourne l'entier congru à nb1 modulo nb2, cet entier étant pris : . Dans {0,...,nb2-1} si nb2>0
. Dans {nb2+1,...,0} si nb2<0

1+
format: (1+ r)

Retourne r + 1.

```
        Ex :        (1+ 4) ==> 5
```

-1+
format: (-1+ r)

Retourne r - 1.

```
        Ex :        (-1+ 4) ==> 3
```

abs
format: (abs r)

Retourne la valeur absolue de r.

Les fonctions suivantes existent aussi et on la même syntaxe que abs : sqrt, sin, cos, tan, exp, log (c'est le log népérien!), acos, asin, atan.

A noter : Quelques fonctions qui peuvent être pratiques ou dont la syntaxe a été étendue :

```
(atan r1 r2) ==> atan(r1/r2)
(log r1 r2) ==> logarithme de base r2 du nombre réel r1.
(lcm nb1 ... nbn) ==> PPCM
(gcd nb1 ... nbn) ==> PGCD
(min r1 ... rn) ==> retourne le plus petit de r1 ... rn
(max r1 ... rn) ==> retourne le plus grand de r1 ... rn
(expt r1 r2) ==> retourne r1 puissance r2 (r1^r2)
```

Scheme, Expressions et procedures

Arthur Accroc

III) EXPRESSIONS ET PROCEDURES :

En Scheme, les programmes sont des expressions-lambdas (c'est à dire un objet qui retourne #<procédures>) évaluées pour retourner des résultats. Les programmes sont tous des fonctions (la plupart du temps récursive). Nous devons maintenant apprendre à évaluer les expressions de façon plus formelle.

1) règles d'évaluation des expressions :

L'évaluation des expressions est soumise à des règles précises ; elles sont (du moins pour l'instant :->) au nombre de cinq :

REGLE 1 : Evaluation des nombres

Les nombres s'évaluent en eux-mêmes (normal!)

REGLE 2 : Evaluation des appels de primitives

(on appellera primitive les procédures 'fournies' avec le Scheme telles que +, -, etc...)

Pour évaluer un appel de primitive, évaluer d'abord chaque argument puis appliquer la primitive aux valeurs résultantes.

Ex : pour évaluer (+ 5 (* 2 10)), on évalue d'abord 5 et (* 2 10); 5 étant un nombre, il s'évalue en lui même selon la règle 1 ; pour évaluer (* 2 10), on aplique 2 fois la règle 1 (pour 2 et 10) et on évalue * ce qui nous donne 20. Une fois arrivé à (+ 5 20), on évalue la fonction + ce qui nous donne 25.

REGLE 3 : Evaluation des expressions commençant par une forme spéciale

Pour évaluer une expression composée commençant par une forme spéciale, il ne faut pas utiliser la règle 2, non, jamais, c'est très MAL!!! (ahhhhhhrrrrg, je m'emporte !) . Chaque forme spéciale a sa propre et unique règle d'évaluation... c'est d'ailleurs pourquoi elles sont appelées formes spéciales !

Nous connaissons pour l'instant deux formes spéciales : DEFINE et LAMBDA (creusez-vous un peu les méninges pour comprendre pourquoi ou regardez quelques dizaines de lignes plus bas...).

REGLE 4 : Evaluation des symboles

Les symboles peuvent être attachés (synonyme de liés) à des objets de données (des nombres par exemple, comme nous l'avons vu pour a quelques lignes plus haut). Pour évaluer un symbole, regarder s'il est lié à un objet de données. S'il en est ainsi, retourner l'objet, sinon signaler une erreur.

```
                    Ex : (define a 4)
                         (define b 5)
                         (+ a b) ==> pour évaluer ceci,    on remplace a
et b par
                         les objets auxquels ils sont liés, ce qui donne
                         (+ 4 5) ==> 9
```

(Attention, quand je dis objet, il ne s'agit pas de POO !!!)

REGLE 5 : Evaluation des appels aux procédures liées

(on appelle procédures liées les procédures qui ne sont pas des primitives, regardez f au sous-chapitre suivant...)

Pour évaluer un appel à une procédure liée, évaluer tout d'abord les opérandes de la procédure. Evaluer alors le corps de la procédure en remplaçant les arguments (c'est à dire ce qui suit le lambda) par les résultats de l'évaluation des opérandes. Le résultat de la procédure (c'est à dire ce qui est retourné), est la valeur de la dernière expression dans le corps de la procédure.

Ex : Cf sous-chapitre suivant !

Voila, avec ces règles, vous pouvez évaluer n'importe quelle expression SCHEME. (Bien entendu à condition de connaître les formes spéciales...qui seront détaillées une par une tout au long de l'initiation)

2) La création de procédures :

DEFINE crée un lien entre deux objets :

Par exemple, (DEFINE a 4) crée un lien entre a et 4. Cela signifie qu'après la définition de a, (+ a 2) pourra être évalué et retournera 6.

De même, on peut définir des fonctions : nous voulons par exemple définir la fonction f qui à x et y associe (x+y)/(x-y). Regardons tout d'abord la notation de cette fonction en lambda-notation ; ça va nous donner

```
(lambda (x y) (/ (+ x y) (- x y)))      (qui est un objet !!)
```

Maintenant, définissons la fonction f:

```
(DEFINE f (lambda (x y) (/ (+ x y) (- x y))))
```

Le PCS retourne F. Cela signifie qu'un lien a été établi avec F (de même que pour A plus tôt). Si on évalue F, PCS retourne #<PROCEDURE>. Maintenant que f est définie, nous pouvons l'utiliser à la manière d'une primitive : (f 1 2) retournera -3. Ceci découle de la règle 5 : En effet, l'évaluation de (f 1 2) entraîne l'évaluation de :

```
((lambda (x y)    (/    (+ x y) (- x y))) 1 2)
```

car 1 et 2 s'évaluent en eux-même selon la règle 1 ce qui entraîne l'évaluation de (/ (+ 1 2) (- 1 2)) qui retourne -3.

REMARQUE : Define est une forme spéciale car pour évaluer une expression contenant DEFINE, PCS utilise des règles spéciales; Je m'explique : Quand PCS évalue (define f (...)),il devrait d'abord évaluer les arguments de la fonction define avant d'appliquer define (selon la Règle 2); en effet, par exemple dans (+ (sqrt 4) 5), PCS évalue sqrt avant + ; or, pour DEFINE, la règle est différente car il n'évalue pas les deux arguments de define avant d'appliquer define (et une chance, car PCS aurait bien du mal à évaluer f vu qu'elle n'a pas encore été définie !!). C'est pour ça que l'on dit de DEFINE que c'est une forme spéciale.

Détail amusant : essayez de taper (define + *) puis (+ 3 2)... hé oui, on peut redéfinir les primitives... ce que je vous déconseille fortement de faire sous peine d'avoir des surprises...

REMARQUE IMPORTANTE : En Scheme, il est possible d'abréger la lourde notation de la forme define en virant le lambda ; voici la nouvelle syntaxe équivalente à la précédente mais plus lisible (et qui sera utilisée par la suite).

```
(DEFINE (fonction var1 var2 ... varn) (corps de la fonction))
```

Ainsi,

```
(define carre (lambda (x) (* x x))
```

peut être remplacée par

```
(define (carre x) (* x x)).
```

De même, pour reprendre l'exemple de f définie plus haut, on aurait pu écrire

```
(define (f a b) (/ (+ a b) (- a b)))
```

Init.au Scheme, Faire des choix

Arthur Accroc

IV) FAIRE DES CHOIX :

Comme dans tout langage, il est possible de poser des aiguillages dans le flux du programme, et ceci grâce aux formes spéciales IF et COND. L'utilisation de ces formes spéciales impose l'utilisation des expressions booléennes.

1) Les expressions booléennes :

En Scheme, vrai s'écrit #T (comme "true" en anglais) et faux s'écrit indifférament #F (comme "false" en anglais) ou (), symbole de la liste vide. Pour l'instant, vu que nous n'avons pas encore vu les listes, nous écrirons faux selon la notation #F (même si PCS l'évalue en retournant la liste vide !).

Les expressions booléennes sont de la même forme que dans les autres langages si ce n'est qu'elles suivent elles aussi les règles de la notation préfixée; En voici les opérateurs :

```
=              egalité numérique
<              strictement inférieur
>              strictement supérieur
<=            inférieur ou égal
>=            supérieur ou égal
<>            différent
AND          ET logique
OR            OU logique
NOT          NON logique
XOR          OU EXCLUSIF logique
```

```
Ex : (= 1 2) ==> ()
          (< 1 2) ==> #T
          (AND (= 1 2) (< 1 2)) ==> ()
```

Il faut rajouter à cette liste quelques primitives booléennes :

EQ? retourne #T si les deux arguments sont liés au même objet.

EQUAL? retourne #T si les deux arguments retournent la même valeur quand ils sont évalués.

ZERO? retourne #T si son argument vaut 0

ODD? retourne #T si son argument est impair, () sinon

EVEN? retourne #T si son argument est pair, () sinon

Les fonctions ATOM?, PAIR?, NULL?, LIST?, etc... seront présentées la prochaine fois (quand nous aborderons les listes).

NOTE : La différence entre eq? et equal? apparaîtra dans les prochains articles lorsque l'on parlera des listes... Pour l'instant, ils peuvent être considérés comme identiques.

RE-NOTE : Maintenant que nous commençons à nous familiariser avec la syntaxe du scheme, nous allons adopter une notation plus simple pour les exemples la première ligne est ce que l'on tape et la deuxième ce que l'interpréteur retourne.

```
[1] (define a 4)
A
[2] (define b 4)
B
[3] (define c 0)
C
[4] (equal? a b)
#T
[5] (zero? a)
()
[6] (zero? c)
#T
```

2) La forme spéciale IF :

format : (IF condition action_si_vrai action_si_faux)

Par exemple, (if (> 2 1) 45 65) retournera 45 car (> 2 1) est vrai.

(IF est la 'version Scheme' du IF THEN ELSE classique!)

3) La forme spéciale COND :

```
        format : (COND (cond_1 action_1)
                       (cond_2 action_2)
                       .....
                       (cond_n action_n)
                       (else action_alternative) )          (<-- facultatif)

        ou encore

                      (COND (cond_1 action_1)
                          (cond_2 action_2)
                          .....
                          (cond_n action_n)
                          (#T action_alternative) )              (<--
facultatif)
```

Dès qu'une condition est remplie, on sort du bloc COND.

Par exemple, le IF du haut peut être transformé ainsi :

(cond ((> 2 1) 45) (else 65)) et retournera 45.

COND est la 'version Scheme' du CASE classique!

NOTE : else et #T sont ici équivalents... en effet, la ligne de condition commençant par #T sera exécutée car #T renvoie #T, autrement dit, la condition #T est toujours vérifiée.

REMARQUE : Dès qu'une des clauses du COND est vérifiée, on sort du bloc COND en retournant la valeur correspondant à la clause vérifiée; Il ne faut donc pas commencer par une clause de type else! Par ailleurs, si l'ensemble de vos clauses ne forment pas une tautologie, il faut classer les clauses dans un ordre décroissant de priorité (c'est logique mais ça va mieux en le précisant!)

4) Les fonctions booléennes :

Il est possible maintenant de définir des fonctions qui renverront un booléen. Voici un exemple :

Ecrire une fonction qui renvoie #T si un nombre x appartient à [a,b[

La manière à laquelle on pense en premier est la suivante :

```
            (define (appartient? x a b) (if (and (<= a x) (< x b)) ;condition
                                            #T                    ;si oui, retourne
#T
                                            #F))                ;sinon, retourne #F
```

Mais cette fonction peut s'écrire de manière plus élégante de la manière suivante :

```
      (define (appartient? x a b) (and (<= a x) (< x b)))
```

utilisation :

```
      (appartient? 4 1 6)
      #T
      (appartient? 7 1 6)
      ()
```

Petit rappel pour les non-matheux : facto(n)=1*2*...*n-1*n=n!

Bon, d'accord, ce n'est pas tout à fait exact, mais disons plutôt qu'il est extrèmement "inélégant" de faire de l'itératif dans les langages de programmation fonctionnelle...

De toute façon, ne comptez pas sur moi pour vous donner les structures itératives du Scheme (je n'en ai d'ailleurs jamais parlé ;-)

Init.au Scheme, Quelques exemples

Arthur Accroc

V) QUELQUES EXEMPLES PRATIQUES (enfin...)

1) Démonstration de la puissance du Scheme en arithmétique entière :

Nous allons définir la fonction factorielle de manière récursive ... de toute façon, on n'a pas le choix vu qu'il n'existe aucune structure de boucles en Scheme.

```
(define (facto n)
        (if (= 1 n) 1 (* n (facto (-1+ n)))))
```

et voila la fonction factorielle définie !

Essayez par exemple (facto 3), ça retourne 6, ce qui est normal ; N'essayez pas facto avec des réels ou des entiers négatifs car la fonction ne s'arrêterait jamais, la condition d'arrêt (= 1 n) n'étant jamais remplie.

NOTE : Si jamais PCS se bloque lors de l'exécution d'un programme, la seule façon de l'arrêter proprement est le [CTRL]+Pause (<==> Attn).

Les aficionados du Pascal et autres C me diront : " Mais on fait la même chose!!! " ; Et moi de leur répondre : "essayez-donc de calculer 1995! avec votre langage... " ; eux : " oulala, ma machine est toute saturée!!! "

Trêve de plaisanteries, essayez donc ceci en Scheme...

Ca y est ? Etonné(e), non ! PCS vient de vous donner le résultat en multi-précision sur la longueur de 2 ou 3 pages écran... Les acharnés me répondront qu'il est facile d'arriver au même résultat en C en stockant par exemple les chiffres dans des tableaux, mais c'est tout de même beaucoup moins immédiat!

NOTE : Nous avons abordé ici le concept de récursivité que nous développerons dès le prochain article... (à moins qu'un ROR ne fasse un joli article sur la récursivité, ce qui m'épargnerait pas mal de travail :-)

2) De l'insistance sur le fait que PCS peut faire ce que ne fait pas PASCAL ou C (ou alors au prix des pires contorsions):

Regardez ce programme :

```
(define (test x)
            ( (if (>= x 0) + -) 10 x )))
```

Le fonctionnement de ce programme n'est pas trivial : son comportement varie en fonction de la donnée : en effet, la section IF va renvoyer une procédure différente (+ ou -) selon la valeur de x.

- si x est positif, (+ 10 x) sera évalué, la section IF ayant été évaluée en +
- si x est négatif, (- 10 x) sera évalué, la section IF ayant été évaluée en -

... étonnant et à mon avis infaisable en PASCAL (du moins de façon si courte et si élégante...quoique)

Initiation au Scheme I, Conclusion

Arthur Accroc

CONCLUSION (provisoire) :

Nous avons maintenant appris les bases du Scheme ; Cependant, il reste à voir de nombreuses choses : Je pense aborder dès l'article prochain les listes (vu que c'est tout de même le principal intérêt des LISP!) et surtout approfondir la notion de récursivité, chose indispensable et inévitable dans ce type de programmation...

Dans les articles suivants (une fois que les bases seront bien établies, autrement dit dans un ou deux articles...), je pourrai selon la demande (quelqu'un serait-il parvenu jusqu'à ces lignes???) et mon humeur parler des structures de données abstraites (matrices, multi-ensembles, arbres, etc...) ou encore proposer quelques applications élémentaires et simplistes du Scheme à l'I.A. .

J'espère que ce (très) maigre aperçu de ce langage vous aura mis en appétit, ou tout du moins intrigué ("hé, vous, au fond, mettez la main devant votre bouche quand vous baillez !"). Pour toute question, je reste à votre entière disposition par courrier électronique (rapide mais pour les réponses courtes) ou ordinaire (lent mais plus pratique pour les questions nécessitant de longues réponses !)

Encore une précision : ne prenez pas cet article comme une déclaration de guerre aux autres langages ! La preuve en est, je programme la plupart de temps en Pascal en bon VBQ (Véritable Bouffeur de Quiche, Cf l'article 'Etes-vous un vrai programmeur?' du rep n°4) tout simplement parceque, et la je vais être dur, mais honnête, je ne pense pas qu'il soit possible de TOUT programmer en scheme (même si c'est théoriquement possible, ça ne sera sûrement pas toujours très 'adapté', ça le sera même plutôt rarement...)

Pour finir, j'ai tenu à vous proposer quelques petits problèmes en annexe pour vous permettre de vérifier si vous avez bien compris ce que j'ai essayé d'exposer... (je pense qu'ils sont très simples mais bon...)

Réponses dans le prochain article...

```
D'ici là, (define (mot_de_la_fin lecteur)
                    (if (equal? lecteur lecteur_satisfait)
                      a_bientot
                      adieu!)
```

-=≡ A. ACCROC ≡=-

ANNEXES

Initiation au Scheme (I), Annexes

Arthur Accroc

ANNEXES

A. PROBLEMES

1- Problèmes simples :

1.1 : Que font ces fonctions ? (define (mystere1 n) (expt n (/ 1 3))) (define (mystere2 n) (* 3.14159 2 n)) (define (mystere3 n1 n2) (/ (+ n1 n2) 2))

1.2 : Ecrire un programme proche1? Prenant deux nombres comme argument et retournant #T si l'écart entre les deux est inférieur à 0.01. Modifier ce programme de façon à pouvoir choisir la précision.
NOTE : Ce programme peut être écrit sans utiliser IF ou COND... Si vous avez bien compris la programmation fonctionnelle, les fonctions booléennes doivent suffir !!

2- Problèmes un peu plus complexes

2.1 : Ecrire un programme calculant le PGCD de deux nombres en utilisant par exemple l'algorithme d'Euclide... Bien évidemment, il ne faut pas utiliser la primitive GCD !

2.2 : Ecrire un programme vol_cone qui prend comme argument la hauteur H et le rayon R de la surface de base du cône et qui renvoie son volume (oui, ça, c'est facile!). Puis écrire un programme vol_cone_h qui prend comme argument la hauteur du cône et QUI RETOURNE UNE FONCTION (hihi, je vous l'avais bien dit en introduction...) prenant comme argument la surface de base et retourne son volume.
TRUC : utiliser vol_cone dans vol_cone_h ainsi que la forme spéciale lambda.

Une idée serait de mettre les réponses dans ma BAL sur TEASER pour que je vois si j'ai tapé trop haut / trop bas et que je puisse rectifier le tir dans le prochain article...

B. BIBLIOGRAPHIE

Il existe énormément de livres traitant de la programmation en LISP, dont quand même pas mal au sujet du Scheme en particulier... Malheureusement, 99% de ces livres sont en anglais. Je peux cependant vous recommander les suivants :

-**PROGRAMMING IN SCHEME**de Michael Eisenberg (Ed. Harold Abelson) sur lequel est fondé une partie de cet article ainsi que les problèmes faciles. Bien évidemment entièrement dans la langue du grand cheik Spirrh...

-**Revised Report on the Algorithmic Language Scheme**par Abelson et Sussman édité par Jonathan Reeves et William Clinger [1986] : ce livre contient la définition langage Scheme par ses créateurs.

C. LANGAGE

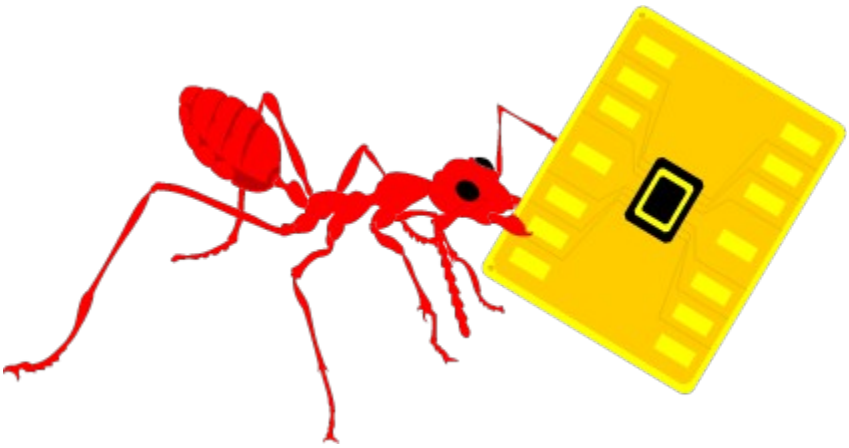
PCS est FREEWARE ainsi que le SCHEME fourni avec Linux. Je les possède les deux ; Que ceux qui le veulent m'envoient une enveloppe timbrée à leur adresse en précisant ce qu'ils veulent sans oublier les disquettes formatées ! (1 pour PCS... Pour la version Linux, me contacter).

Par ailleurs, PCS sera peut-être en téléchargement sur TEASER à l'heure ou vous lisez ces lignes, tout cela dépendant la la décision de CHIP... en effet, l'archive ARJ (et en version NON auto-extractible !) fait quand même environ 460 Ko et ça risque d'être assez laborieux à télécharger...

Encore là??? Mais puisque je vous dis que c'est fini ! ;-)

Tri de fourmis

Dionys



MODELISATION INFORMATIQUE ou
Le nouvel essor des sciences grâce à l'ordinateur

DIONYS

MODELISATION INFORMATIQUE ou
Le nouvel essor des sciences grâce à l'ordinateur

PREFACE

La préface qui suit est un petit discours sur la modélisation informatique. Si ça vous rase, n'hésitez pas à la sauter et à passer directement au sujet principal: "L'intelligence collective" et son exemple à travers la société des fourmis, au chapitre 1.

Je voudrais faire ici une petite introduction sur la modélisation informatique (représentation sur ordinateur d'un modèle théorique), ce qu'elle représente, à quoi sert-elle et sa puissance par rapport à d'autres moyens d'expérimentation. Mais comme je ne suis pas un professionnel de cette technique, je laisserai parler la plupart du temps Heinz Pagels, par l'intermédiaire d'extraits de son livre (dont je ne me rappelle hélas plus le nom), livre très intéressant sur les sciences en général et les nouvelles sciences qui s'appuient sur l'ordinateur.

Le scientifique Heinz Pagels raconte comment il s'est rendu compte de l'importance de l'ordinateur dans les expérimentations et la modélisation:

> **"J'ai commencé à mesurer la portée de l'événement en assistant à un séminaire donné par l'un de mes collègues physicien théoricien. Décrivant sa théorie des particules quantiques, il cita une "expérience" venant étayer cette théorie. Le problème était que je ne pouvais pas imaginer que quelqu'un puisse faire cette expérience (elle dépassait de beaucoup nos capacités expérimentales). Je compris alors que mon collègue parlait d'une "expérience" sur ordinateur grâce à laquelle il avait modélisé une collision de particules quantiques."**

> **"On peut très bien considérer la nature comme un calculateur analogique. La science future progressera entre autres en combinant des observations de systèmes réels et leur modélisation par ordinateur. Cela diffère de la notion traditionnelle d'expérimentation, qui suppose que l'on modifie activement les conditions du système réel pour en comprendre le fonctionnement. Pour beaucoup de systèmes naturels réels, comme l'intérieur des étoiles, on ne peut même pas faire d'expériences, et la modélisation est la seule voie que l'on puisse suivre. [...] La modélisation sur ordinateur offre un nouveau moyen puissant d'investigation. La modélisation sur ordinateur est un nouveau type d'expérimentation."**

En résumé, on remarque désormais les limites de l'expérimentation classique qui ne peut pas s'appliquer sur de nombreux systèmes réels. La science avançant, elle repousse de plus en plus les limites de l'inconnu mais aussi de l'accessible. Pour pallier à ce manque, les chercheurs ont désormais recours à l'ordinateur: l'observation du système réel puis sa modélisation (sa représentation) sur ordinateur leur apporte un nouveau type d'expérimentation.

"Comme le microscope et le télescope l'ont fait jadis, l'ordinateur ouvre une nouvelle fenêtre sur la réalité. Ou crée-t-il en fait cette réalité? Comment pouvons nous donc dire si la réalité révélée par des simulations par ordinateur n'est pas seulement un objet fabriqué, un produit de l'ordinateur? Il est indéniable que les données que l'on entre et le programme sont des objets fabriqués par l'homme. Les ordinateurs sont en fin de compte des machines "non intelligentes", tout comme les télescopes et les microscopes, et nous ne pouvons pas nous passer du jugement humain [...] pour décider si oui ou non une simulation est correcte. Nous devons donc considérer l'ordinateur comme un instrument scientifique aux mains de l'homme et non comme une "boîte noire magique" qui créerait une réalité dépassant notre entendement. Sinon, nous ne manquerons pas de confondre la simulation avec la réalité."

Ce paragraphe sur l'utilité de l'ordinateur est à mon sens le plus important. Je me posais en effet la question que tout le monde se pose: mais comment peut-on être sûr que ce que représente l'ordinateur correspond bien à la réalité (si tout au moins cette réalité peut exister)? La façon dont Heinz Pagels nous présente l'ordinateur, ce nouvel instrument dans les mains du scientifique, comme naguère il utilisait le microscope, me (nous?) rassure un peu plus: ce qu'offre l'ordinateur sera toujours jugé en dernier lieu par l'esprit humain. A l'homme de décider si ce qu'il voit lui parait correct.

"Par la modélisation par ordinateur, les scientifiques essaient généralement de modéliser un système plutôt complexe. L'hypothèse de base derrière la simulation de systèmes complexes est que l'apparente complexité du système que l'on tente de modéliser est due à quelques composantes simples interagissant selon des règles simples qui sont alors incorporées dans le programme. Dans un certain sens, tout en étant très réelle, la complexité de certains systèmes a une explication simple."

Faire du complexe avec du simple, c'est maintenant un phénomène bien connu: les harmoniques (superposition d'oscillations sinusoïdales pour former une oscillation complexe), la nouvelle théorie du chaos ("le chaos est l'art de former du complexe à partir du simple", dixit Pierre Bergé et Yves Pommeau dans la préface du Hors Série de "Pour la Science" sur le Chaos) ou même, en programmation, la fameuse "programmation structurée" développée par Niklaus Wirth et dont le Pascal en est la référence. Tous ces exemples montrent bien qu'un problème se présentant sous une apparente complexité peut en fait se décomposer en une série de sous-problèmes plus simple; le tout étant de trouver ces sous-problèmes.

Avant de passer au vif du sujet, je tenais juste à vous citer un dernier exemple de l'importance de la modélisation informatique. Vous êtes certainement au courant que de nombreuses nations ont décidé d'arrêter les essais nucléaires à la fin 1996. Pourtant, ces nations continueront à réaliser des essais, mais cette fois-ci, ils les feront dans leur pays, au beau milieu de la population. Bien évidemment, ces essais ne seront pas en grandeur nature, mais ils seront "simulés" sur ordinateur. Comme quoi l'ordinateur peut contribuer au développement de la paix...

Passons maintenant au Chapitre I

Citations extraites de : "Les rêves de la raison" (L'ordinateur et les sciences de la complexité) Heinz PAGELS, trad. Michèle GORENNE -- INTEREDITIONS

Chapitre I, Tri fourmi, Dionys

DIONYS

MODELISATION INFORMATIQUE ou
Le nouvel essor des sciences grâce à l'ordinateur

I Introduction à l'intelligence collective

L'article qui suit prend ses sources dans celui écrit dans le mensuel "Pour la Science n 198, Avril 1994" (édition française de "Scientific American") par Guy Theraulaz, Eric Bonabeau, Simon Goss et Jean-Louis Deneubourg. L'article était clair, précis et suffisamment complet pour entrevoir les possibilités de la modélisation sur ordinateur.

Avez vous déjà lu le livre **"Les fourmis" de Bernard Werber**? Après cette lecture, on regarde d'une autre façon la petite bestiole à 6 pattes: comment une si petite "chose" peut-elle construire des nids aussi complexes, des villes entières avec ses hiérarchies? Comment peuvent elles réaliser une société avec ses guerres, ses grands travaux, etc...? Quels sont les mécanismes qui permettent à chaque individu de l'espèce de régler ses activités de façon à favoriser la survie de son espèce?

Vu d'en haut (c'est à dire de nos yeux), on a l'impression qu'un agent invisible coordonne toutes les activités de ces insectes: la reine pourrait-elle ainsi agir sur le comportement de chacun? Il est clair que cela parait peu probable, vu le nombre incroyable de fourmis se côtoyant dans un nid.

Vers 1950, des membres de l'école d'entomologie française parièrent plutôt (et montrèrent) sur l'idée que ce comportement global résulte d'une multitude d'interactions locales (ou individuelles). Pierre-Paul Crassé, de cette école, développa notamment la théorie de la "stigmergie" pour expliquer la coordination de ces tâches individuelles. Un exemple pour expliquer ce terme barbare est la construction d'un nid de termites: l'évolution de ce nid ne dépend pas directement des ouvriers mais des constructions elles-mêmes. L'ouvrier ne dirige pas son travail, il est guidé par lui: au fur et à mesure de la construction du nid, l'ouvrier termite reçoit des stimuli qui modifient son comportement selon l'évolution du travail.

Un des auteurs de l'article qui me sert de référence (Jean-Louis Deneubourg) se rendit compte que ces phénomènes de coopérations entre insectes correspondent à des phénomènes d'auto-organisation qui résultent de communications entre les individus d'une part, et des interactions des insectes et de leur environnement d'autre part.

Rentrons maintenant dans le vif du sujet, c'est à dire le but du programme. A l'intérieur d'un nid de fourmis, on remarque que les divers éléments du couvain (les oeufs, les larves, les cocons...) sont regroupés par paquets. C'est le travail des ouvrières que de réaliser ce tri. Mais comment font-elles pour s'entendre entre elles et ne pas se gêner les unes des autres? Un exemple de ce phénomène est de renverser le contenu d'un nid sur une table. Très vite, vous allez voir les ouvrières s'affairer à trier ce nid de nouveau, **SANSCOMMUNICATION DIRECTE ENTRE ELLES**. Et pourtant, malgré un total désordre apparent, le nid va finir par se retrouver trié.

Le but de ce programme est de modéliser ce comportement des ouvrières lorsqu'elles trient leur nid, ou plus précisément de montrer qu'un comportement général (celui du nid) peut résulter d'une série d'interactions de comportements personnels (celui d'une fourmi).

Voyons voir comment on peut modéliser ce tri à l'aide du Chapitre II

Chapitre II, Tri fourmi, Dionys

DIONYS

MODELISATION INFORMATIQUE ou
Le nouvel essor des sciences grâce à l'ordinateur

II Modélisation du problème

Tout d'abord, voyons les problèmes purement informatiques: comment sera représenté le nid?

◆ Le nid en lui-même sera représenté par l'écran: les bords de l'écran représenteront les limites du nid. On voit déjà que l'on perd une dimension au problème: les fourmis se déplaceront dans un espace à 2 dimensions, et non 3 comme dans la réalité.

◆ Les fourmis seront représentées par un caractère spécial (le "&") à l'écran (nous sommes en mode texte).

◆ Des caractères colorés (un espace effectué avec une couleur de fond) représenteront les objets à trier. 2 couleurs différentes correspondront à 2 objets différents.

a) Modélisation

Regardons maintenant ce que l'on désire réellement faire, c'est à dire le problème modélisé (décrit de façon mathématique) mais pas encore informatisé.

Prenons une fourmi: elle se déplace aléatoirement sur notre nid en 2 dimensions. Elle rencontre soudain un objet. Elle va alors le prendre sur son dos et le transporter si l'objet se trouve isolé. Elle risque donc de s'emparer d'un objet avec une probabilité d'autant plus grande que l'objet est isolé. Du coup, tous les objets plus ou moins isolés sont recueillis.

Imaginons maintenant que cette fourmi ait embarqué cet objet. Elle continue alors son "bonhomme" de chemin avec son objet sur le dos. Quand va t'elle le redéposer? Hé bien, si elle veut trier le nid, elle ne va pas le déposer n'importe où mais là où d'autres objets de ce type sont déjà entreposés. Elle risque donc de déposer son objet avec une probabilité d'autant plus grande qu'il y a déjà des objets de ce type dans son voisinage. De cette façon, des petits tas d'un même type d'objet risquent de se créer.

b) Approche informatique

Relisons maintenant le modèle précédent avec nos yeux d'informaticien.

◆ "La fourmi se déplace aléatoirement sur le nid": en programmation, ça se lit "déplacer notre caractère_fourmi à l'écran, à chaque unité de temps, d'un caractère, dans une direction aléatoire (vers le haut, vers le bas, vers la gauche ou vers la droite)."

◆ "Elle rencontre un objet": cela signifie donc que notre caractère_fourmi se trouve sur une case colorée (différente de la couleur du fond = couleur du nid).

◆ "Elle s'empare de l'objet avec une probabilité d'autant plus grande que l'objet est isolé." Aïe, des probas! En plus notre fourmi doit pouvoir estimer "l'isolement" de l'objet qu'elle a rencontré. Pour une fourmi réelle, elle peut sûrement estimer ça à l'aide de son odorat: la concentration en odeur lui permet de savoir si l'objet est seul ou non. Ou bien alors une exploration tactile lui permet de connaître "le taux d'isolement" de son objet.

Houlà !, nous voilà t'y pas qu'il nous parle d'odorat et de toucher maintenant. Ca va plutôt être dur de faire comprendre ça à un ordinateur !!" Je ne vous le fais pas dire... alors du coup, on donne à notre fourmi informatique une mémoire à court terme qui va lui permettre de se rappeler ses derniers objets rencontrés. Du coup, notre "taux d'isolement" d'un objet est calculable à l'aide de cette mémoire. Imaginons en effet que notre fourmi ait rencontré 3 objets d'un même type lors de ses 15 derniers pas: le taux d'occupation de cet objet sera donc de 3/15. Simple non? Oui bon, d'accord, mais ça ne se rapproche pas vraiment de la réalité ça de donner une mémoire à une fourmi? Oui... mais on s'en fout, du moment que le résultat est le même. N'oubliez pas: on MODELISE le problème.

Nous voilà donc avec notre taux d'occupation. On peut maintenant calculer la probabilité qu'a la fourmi de s'emparer d'un objet, à condition évidemment qu'elle n'en ait pas déjà un sur le dos: Soit f le taux d'occupation d'un objet, on pose p1 la probabilité de s'emparer d'un objet et on a: $p1 = (K1 / (K1+f))^2$ où K1 est une constante. On voit bien que plus l'objet est isolé (f -> 0), plus la fourmi a de chance de s'emparer de cet objet (p1 -> 1).

◆ "Elle dépose son objet avec une probabilité d'autant plus grande qu'il y a déjà des objets de ce type dans le voisinage." Une fois sur une case vide, notre fourmi_informatique doit donc se poser la question suivante: "est-ce que je peux poser mon caractère_objet ici?" La réponse à cette question est donnée par une probabilité, du même style que la précédente: On pose p2 la probabilité de déposer l'objet. On a: $p2 = (f / (K2+f))^2$ avec f notre taux d'occupation de l'objet, et K2 une constante. On remarque que plus la concentration de l'objet est forte (f -> 1), plus la fourmi a une chance de déposer son objet.

◆ Dernier point: on parle de probabilité faible ou forte, ce qui n'est pas très "parlant" pour un ordinateur. Prenons le cas de la probabilité p1 qui permet de savoir si la fourmi va s'emparer de l'objet ou pas. On utilise pour ça une fonction aléatoire (Random en Pascal): si Random est supérieur à p1, alors la fourmi s'empare de l'objet, sinon elle continue son chemin. En pseudo-code:

```
si Random>p1
alors prendre-objet
sinon continuer
```

c) Le programme

Bien, maintenant que nous avons vu comment un ordinateur peut comprendre le problème et comment il peut le représenter, nous pouvons lui décrire ça dans un langage qu'il comprendra: un programme. C'est la partie la moins compliquée, il suffit d'appliquer tout ce qu'on a dit précédemment. Ce n'est d'ailleurs pas la peine de lire

ce qui suit si vous n'avez pas envie de vous plonger dans les méandres de la programmation. Du moment que les chapitres précédents ont été compris.

Le langage que j'utilise est le Turbo Pascal à partir de la version 6.00 car il comprend une part de Programmation Orientée Objet (POO), mais tout autre langage peut être utilisé, même sans POO bien que cela ne soit pas très pratique. Je m'explique : la POO est bien utile ici car pour nous, une fourmi sera représentée par un objet (ou plutôt une instance de l'objet fourmi, mais là on s'en fout du vocabulaire utilisé) et elle aura ses caractéristiques et ses actions possibles. Mais tout cela sera expliqué plus précisément par la suite.

Tout d'abord, **le nid**. On a dit qu'il sera représenté par **l'écran**. On dispose donc aléatoirement à l'écran toute une série de caractères de couleurs (un espace coloré) qui représenterontles objets à trier. C'est la procédure **InitNid**qui s'occupe de ça.

Ensuite **les fourmis**. Elles sont créées dynamiquement pour que vous puissiez choisir exactement le nombre de fourmis que vous désirez. Décrivons maintenant les propriétés (les champs) et les actions (les méthodes) de notre fourmi_informatique:

Tout d'abord, les champs:

- **la fameuse mémoire.**Ce sera un tableau qui contiendra les derniers objets rencontrés (la valeur des couleurs des objets). Ce sera le champ **Memoire**.

- **l'objet rencontré.**Ce sera une simple variable qui contiendra la couleur de l'objet (puisque c'est ainsi qu'on repère les différents objets). Ce sera le champ **Objet**.

- si la fourmi a emporté l'objet, elle transporte désormais une charge. Cette **charge**sera représentée par la variable **Charge**qui contiendra, une fois de plus, la couleur de l'objet transporté.

- Pour **repérer**la fourmi sur le nid (à l'écran), on utilise les champs **X**et**Y**qui donnent les coordonnées de la fourmi à l'écran.

De plus, notre fourmi_informatique a les méthodes suivantes:

- En suivant la technique habituelle de la POO, un objet a une méthode **Init**qui s'occupe d'initialiser tous les champs de l'objet. Ici, elle vide la mémoire, positionne la fourmi quelque part sur l'écran et met à 0 d'autres champs.

- Lors de chaque déplacement, elle sauvegarde dans sa mémoire tout ce qu'elle rencontre sur son chemin. C'est le rôle de la méthode **AjouteMem**.

- Notre fourmi_informatique doit pouvoir aussi calculer le **taux d'occupation d'un objet**. La méthode **TxOccup**s'en charge en regardant dans la mémoire. Ainsi, il change en fonction de l'environnement de la fourmi.

- Les principales interrogations que doit se poser la fourmi sont: **"puis-je prendre cet objet devant moi?"**et **"puis-jedéposer ma charge à cet endroit?"**(c'est une fourmi très polie). Pour cela, notre fourmi_informatique utilise les méthodes **Prend**et **Depose**qui vont lui répondre Oui ou Non selon le taux d'occupation de l'objet à prendre ou à déposer.

- D'autres interrogations, d'ordre secondaire, sont: **"est-ceque j'ai rencontré un objet?"**et **"Suis-je en train detransporter un objet?".**Ce sont respectivement les méthodes **IsOccupe**et **IsCharge**qui vont lui répondre.

◆ Mais pour l'instant, notre fourmi_informatique ne **bouge**pas. C'est la méthode **Deplace**qui va l'aider à marcher. Elle ressemblera plutôt à une fourmi ivre mais à une fourmi quand même. En effet, cette méthode va déplacer la fourmi dans un sens aléatoire à chaque fois. De plus, le nid sera considéré comme une Terre en miniature: en effet, le haut du nid correspond au bas, et la gauche correspond à la droite. Ou si vous préférez, lorsque la fourmi dépasse la limite en haut, elle se retrouve en bas et vice versa. De même pour la droite et la gauche. Ce n'était pas obligé de faire ainsi mais je trouvais ça plus marrant, c'est tout.

◆ Enfin, la dernière méthode importante, c'est la méthode **Action**qui s'occupe de contrôler la fourmi: elle déplace la fourmi, puis s'occupe de prendre ou déposer un objet selon ses possibilités.

Maintenant que notre fourmi_informatique est réalisée, nous allons lui donner des petites copines. C'est la procédure **TriFourmis**qui va s'occuper de cela, en déclarant des instances (des copies) de cette fourmi_informatique en mémoire. Puis, il appellera successivement la méthode **Action**de chaque fourmi pour contrôler toutes les fourmis une par une, et ceci en boucle, indéfiniment, jusqu'à ce que vous décidiez d'arrêter le programme.

Maintenant que l'on sait comment ça fonctionne, que devons nous voir? C'est expliqué dans le prochain chapitre, le Chapitre III

Chapitre III, Tri fourmi, Dionys

DIONYS

MODELISATION INFORMATIQUE ou
Le nouvel essor des sciences grâce à l'ordinateur

III QUOI QU'ON DOIT VOIR?

Tout d'abord vous verrez un écran qui vous demandera si vous voulez modifier les données. Pour l'instant, tapez "N" (non) pour garder les données initiales.

Quand je dis données, ce sont:

- le nombre de fourmis
- le nombre d'objets
- le nombre d'objets différents
- les constantes K1 et K2

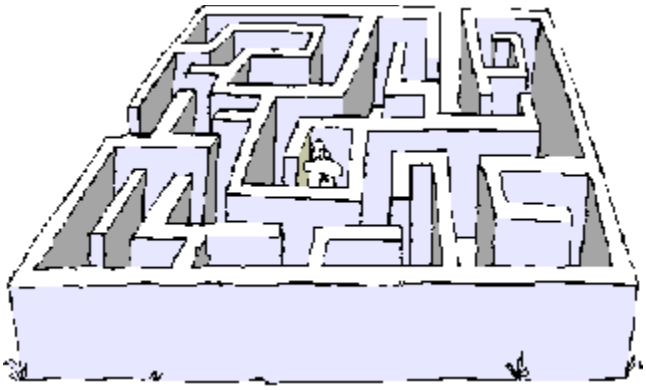
Vous vous retrouvez alors tout de suite devant votre nid modélisé, avec plein de couleurs partout (heu, en fait il n'y en a que 2 mais il y a beaucoup d'objets) et une dizaine de "bestioles" qui bougent partout sur l'écran: ce sont nos fourmis. Comme prévu, le désordre parait total! Ca bouge de partout et nos fourmis paraissent complètement folles (comme dans la réalité, quand vous donnez un coup de pied dans un nid de fourmis. Vous n'avez jamais fait ça? Faut essayer, ça défoule énormément) Mais n'arrêtez pas encore le programme en vous disant "on s'est fait avoir sur la marchandise. Ce Dionys est fou, ça marche pas son truc, etc., etc.. je vous entends déjà. Attendez quelques instants, quelques secondes ("ah tiens, c'est marrant, y a des petits tas qui se forment"), quelques minutes ("ah tiens, c'est encore plus marrant, les petits tas se sont regroupés en tas plus gros") et vous voyez apparaître, dépité (si, si!), des tas d'une même couleur qui se regroupent. C'est gagné !

Amélioration possible: **en fait, dans la réalité, une fourmi peut certainement utiliser son odorat pour se déplacer et donc ne pas avoir un mouvement complétement désordonné. Une amélioration possible de ce programme serait donc de rechercher les objets les plus proche de la fourmi et diriger celle-ci vers ces objets.**

Cliquez ici pour démarrer le programme


Exécuter

Dionys [Jul 95]



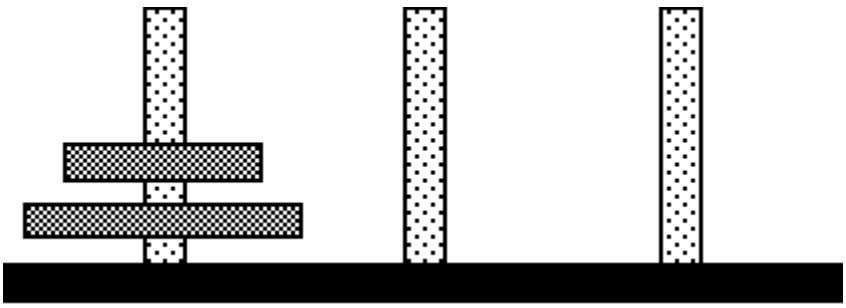
Les Tours de Hanoï

I Présentation

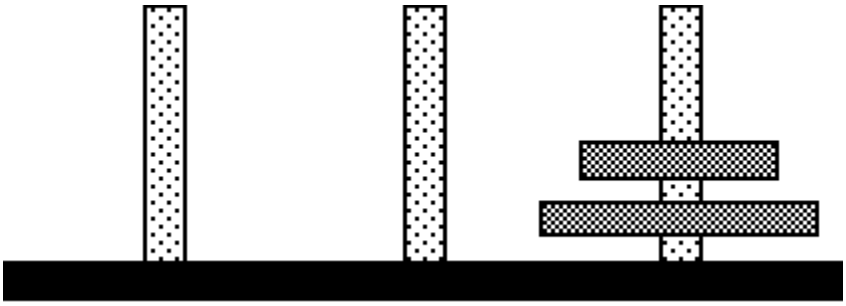
I.1 Règle

Le jeu des tours de Hanoï consiste en une plaquette de bois sur laquelle sont plantées 3 tiges. Sur ces tiges sont enfilés des disques de bois dont les diamètres sont tous différents. La seule règle du jeu est de ne jamais poser un disque sur un disque plus petit que lui. Au début du jeu tous les disques sont posés sur la tige de gauche. Le but du jeu est de déplacer tous les disques (qui sont donc au départ tous enfilés sur la tige de gauche), vers la tige de droite, sans jamais violer la règle.

Situation de départ :



Situation d'arrivée :



I.2 La légende

Le jeu original était accompagné d'une notice racontant la légende de moines d'un temple de Hanoï qui passaient leur temps à résoudre ce jeu pour atteindre le nirvana. En effet, les moines croyaient que la fin du monde arriverait lorsque le jeu serait achevé. Leur jeu grandeur nature occupait la cour d'un temple. Il se composait de 64 disques d'or et de 3 tiges d'ivoire d'un mètre de haut.

Nous allons voir que ce jeu, sous ses allures simples, peut demander pas mal de temps avant d'être résolu. C'est un vrai casse-tête. Mais, alors qu'un être humain mettra du temps à arriver au bout de ce jeu, l'ordinateur va trouver en quelques minutes la solution, simplement en utilisant une règle, bien pratique en informatique et en mathématique : la récursivité.

I.3 Le programme fourni

Mais avant de commencer à étudier le problème, je vous conseille de mettre la main à la pâte en essayant manuellement de résoudre le problème : démarrez le programme puis choisissez l'option "Manuellement". Passez y quelques minutes... avant d'abandonner (si, si, vous abandonnerez. Je ne dis pas que vous ne réussirez pas à trouver mais simplement que vous en aurez vite marre, surtout si vous y allez à tâtons).

Maintenant que vous avez essayé, choisissez l'option automatique pour voir l'ordinateur se débrouiller comme un Dieu, et vous éclater purement et simplement.

Ca dégoûte, hein !!

II Le programme

II.1 Le principe de la récursivité

Il faut dire que ce jeu est une parfaite démonstration de l'utilité de la récursivité. La règle du jeu est simple (ne pas déposer un disque sur un disque plus petit), la solution l'est aussi.

La récursivité consiste à dire : "je suppose que je sais comment faire pour les n-1 premiers cas, comment faire

pour le nième cas ?". Appliquons cela à nos tours.

II.2 L'algorithme

Supposons que les tiges s'appellent A, B, et C, que n soit le nombre de disques, tous posés au départ sur la tige A, et que nous devions les mettre sur la tige C. Appliquons le principe de la récursivité : "je suppose que je sais comment faire pour déplacer les n-1 premiers disques de la tige A vers la tige B, sans violer la règle, comment faire pour déplacer le nième (qui est sur la tige A) vers la tige C, toujours sans violer la règle ?".

Et là, l'illumination vous atteint : Eurêka ! J'ai trouvé ! Eh oui, il suffit simplement de déplacer le dernier disque (donc le plus gros) de la tige A vers la tige C (qui est vide, donc pas de violation de la règle), puis de déplacer les n-1 disques qui sont sur la tige B vers la tige C, par au-dessus, et cela sans violer la règle puisque c'est le plus gros qui se trouve maintenant sur la tige C. Oui, d'accord, mais comment faire pour déplacer les n-1 disques de la tige B vers la tige C ? Eh bien puisque vous avez réussi à les déplacer de la tige A vers la tige B (c'est la supposition que vous avez faite pour le principe de la récursivité, voir plus haut), vous serez capable de les déplacer de la tige B vers la tige C.

Voilà donc le problème résolu pour n disques, en partant du principe que vous savez le résoudre pour n-1 disques.

Mais nous savons aussi résoudre le problème pour 0 disques : il n'y a rien à faire. Nous savons donc résoudre le problème des tours de Hanoï pour tout n.

Voici donc la décomposition à faire, lorsque l'on a n disques :

1. Je transfère les n-1 disques de la tige A vers la tige B
2. Puis je déplace le nième disque de la tige A vers la tige C
3. Je transfère les n-1 disques de la tige B vers la tige C

Aide : pour mieux saisir la récursivité, il faut toujours réfléchir en ignorant totalement comment on fait pour les états précédents. Ca peut paraître paradoxal puisqu'on suppose connus ces états mais c'est ainsi, et ma foi, c'est bien pratique. Ainsi, ici, j'ignore totalement comment il faut faire pour les n-1 disques mais par contre, je sais très bien ce qu'il faut faire pour déplacer le dernier.

On obtient donc l'algorithme suivant:

```
Procédure Hanoï(tige1, tige2, tige3 : caractère; nbdisques : entier)
Début
    Si (nbdisques>0)
        Alors Début
                Hanoï(tige1,tige3,tige2,nbdisques-1)
                Déplace(tige1,tige3)
                Hanoï(tige2,tige1,tige3,nbdisques-1)
        Fin
Fin
```

avec Déplace(de,vers : caractère) une procédure qui déplace le disque posé sur la tige "de" vers la tige "vers".

III Complexité

III.1 Complexité du programme

Déterminons maintenant le nombre de déplacements effectués (le nombre d'appels à la procédure Déplace) lorsqu'on joue avec n disques. Appelons coût(n) ce nombre.

coût(n) est donc le nombre de déplacements effectués lorsqu'on appelle la procédure Hanoï avec n comme argument (Hanoï(x,y,z,n)).

Pénétrons maintenant dans le corps de la procédure :

```
      Si(nbdisques>0)

      /* c'est bon puisque n>0)
*/

      Alors Début

          Hanoï(tige1,tige3,tige2,nbdisques-1)

      /* à    la    suite    de cet appel, il y aura d'autres appels à "déplace" */
      /* mais    combien    ?    Simple,    il    y    en    aura    coût(n-1) car on
joue */
      /* désormais avec n-1 disques
*/

          Déplace(tige1,tige3)

      /* Là c'est clair, il y a 1 déplacement
*/

          Hanoï(tige2,tige1,tige3,nbdisques-1)

      /* Là    pareil,    on    joue    désormais avec n-1 disques, d'où coût(n-1) */
      /* déplacements
*/
```

Ce qui donne pour la procédure de Hanoï:

```
      coût(n) = coût(n-1)+1+coût(n-1)
                      = 2*coût(n-1)+1
```

Et pour 0 disques, c'est simple, il n'y a aucun déplacement à faire, d'où :

```
        coût(n) = 2*coût(n-1)+1
        coût(0) = 0
```

Bien, mais ça ne nous donne toujours pas le nombre de déplacements effectués.

Voyons voir pour n-1, n-2,...

```
coût(n-1) = 2*coût(n-2)+1
coût(n-2) = 2*coût(n-3)+1
...
coût(0)      = 0
```

Ca donne :

```
coût(n) = 2*(2*coût(n-2)+1)+1
                = 2*(2*(2*coût(n-3)+1)+1)+1
                = ....
                = 2*(2*(2*(2*(....(2*coût(0)+1)...)+1)+1)+1)+1
```

Pas vraiment plus clair, il est vrai. Prenons un exemple alors. Prenons n=3 :

```
coût(0)=0
coût(1)=2*coût(0)+1=1
coût(2)=2*coût(1)+1=3
coût(3)=2*coût(2)+1=7
```

Ca a l'air de donner du 2^n-1. On aurait donc coût(n)=2^n-1

Oui, peut-être, mais ça il va falloir le prouver. Et puisque depuis le début on fait de la récursivité, et bien continuons en utilisant la récursivité pour démontrer cela.

C'est tout de même bien pratique la récursivité : si vous savez démontrer que ça marche à 2 endroits, vous savez démontrer que ça marche partout.

Au rang 0 : coût(0)=0 c'est clair, rien à démontrer

Supposons que coût(n)=2^n-1 est vérifiée au rang n. Montrons que cela est aussi vérifiée au rang n+1 (coût(n+1)=2^(n+1)-1 ?)

On a vu que coût(n)=2*coût(n-1)+1 qui est toujours vérifié, donc :

```
coût(n+1) = 2*coût(n)+1
                    = 2*(2^n-1)+1
                    = 2^(n+1)-2+1
                    = 2^(n+1)-1
```

coût(n+1) est donc vérifiée, et par suite du principe de récurrence, coût(n) est aussi vérifiée. Il faut donc 2^n-1 déplacements de disques lorsque l'on joue avec n disques.

III.2 Retour à la légende

Revenons à nos moines avec leur 64 disques. Cherchons combien de temps leur faudra-t'il pour atteindre le nirvana et le temps qu'il nous reste à vivre avant la fin du monde.

Disons qu'il leur faut à peu prés 4 secondes pour déplacer 1 disques (n'oublions pas que leur jeu était en grandeur nature). Ca fait donc 3600/4=900 mouvements par heure. Imaginons qu'ils jouent jour et nuit à ce jeu -> 900*24h = 21600 par jour. Mais ils ont 64 disques, d'où 2^64-1 mouvements, soit ~ 1,8*10^19 mouvements à faire.

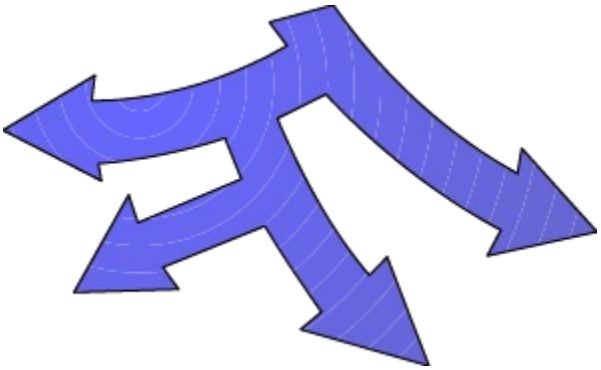
On divise ce nombre par le nombre de mouvements par jour :

1,8*10^19 / 21600 = 8,5*10^14 jours pour finir le jeu.

[Dionys]

Méthodologie informatique

Gillej Jul/Aou 95



Méthodologie informatique par

GILLE Julien
14 Grande Rue
54 170 ALLAIN

B.à.L GILLE J. sur Teaser
Adresse 100657,1745 sur CompuServe
Adresse 100657,1745@compuserve.com sur Internet

SOMMAIRE

1. Methode pour créer un programme informatique

1.1. Introduction

Beaucoup de programmeurs ont commencé à programmer sans aucune méthodologie. C'est mon cas et je sais les problèmes que cela crée. On se retrouve souvent devant un listing de programme confu, où les différentes modifications successives ne sont plus visibles. En fait, on se retrouve devant un programme que l'on ne comprend plus dans sa totalité. Le pire, c'est lorsqu'on veut se replonger dans un programme qu'on a réalisé de nombreuses années (voire quelques mois) auparavant. Suivre une méthode de travail devient alors obligatoire. Différentes méthodes existent, j'adopte celle du "Cycle en V" bien connue des personnes qui suivent des études en informatique.

1.2. Pourquoi le Cycle en V ?

Le "Cycle en V" nous oblige a suivre une démarche très stricte. De plus il souligne une des règles de l'informatique : on ne doit passer qu'environ 40% du temps de la création d'un programme devant son écran. Cela peut paraître dérisoire aux personnes qui ont l'habitude de coder tout de suite ; mais la démarche

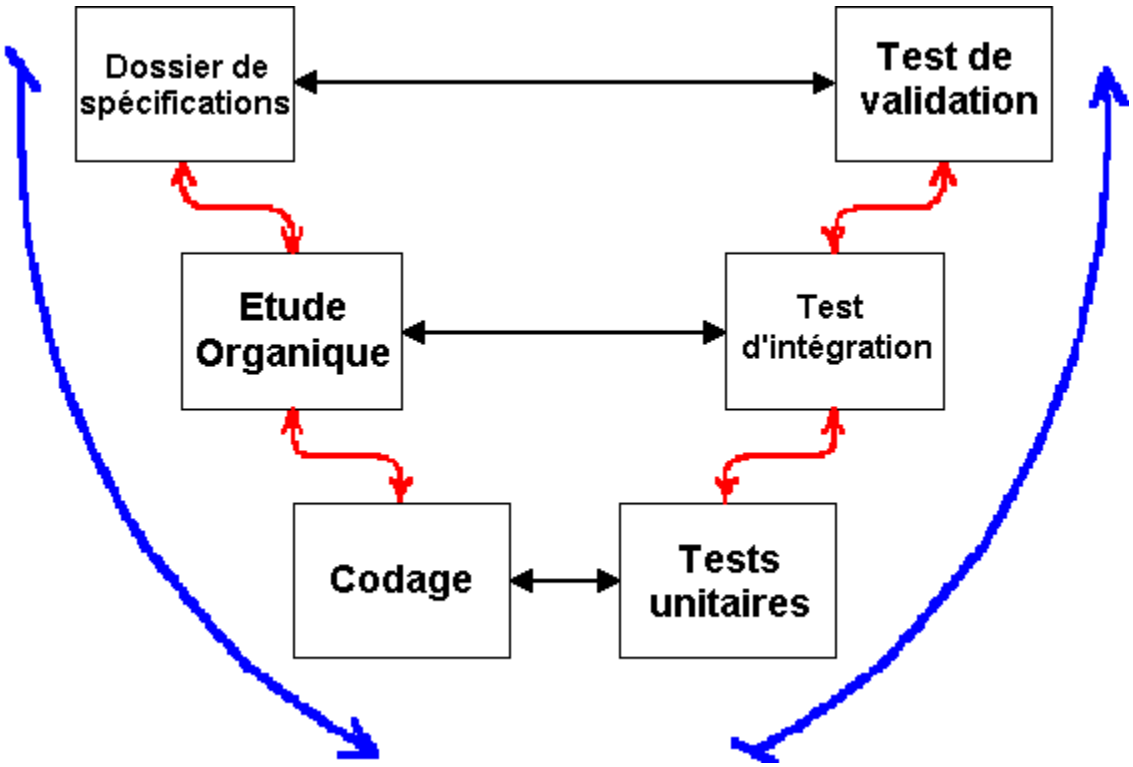
proposée va sûrement vous présenter de nombreux avantages.

2. Le cycle en V

2.1. Introduction

Tout d'abord, une petite remarque :

Je tiens à préciser que je présente ici un "Cycle en V" hybride et synthétique pour pouvoir expliquer beaucoup de notions d'un seul coup. Que les professeurs d'informatique et autres "grosses têtes" ne m'en veuillent pas. Il n'est donc pas la peine de m'écrire pour me dire que ce n'est pas "vraiment ça". A quoi peut bien servir un tel dessin ?



La première fois que l'on m'a présenté ce schéma, je me suis posé la même question. Mais après que me fut expliquée sa signification, ce schéma me devint beaucoup plus évident. C'est ce que je vais tenter de faire. Tout d'abord, il faut savoir ce que l'on veut réaliser dans le programme et ce qu'il doit faire. Tout ceci doit être expliqué dans le cahier des charges.

2.2. Le cahier des charges

Le cahier des charges n'est pas compris dans le cycle en V mais le précède.

Il doit comprendre les attentes d'un client éventuel (Vous pouvez vous placer à la place du client si il n'y en a pas !). Dans ce document, il ne doit pas figurer de spécifications propres au matériel utilisé. Il doit plutôt y être indiqué les options que le programme doit avoir, les entrées qui lui seront fournies et les sorties qu'il devra donner. On peut également indiquer des parties de l'interface graphique désirée si elle doit être figée.

2.3. Le dossier de spécifications

Ah...

Nous voilà enfin dans le "Cycle en V" ! Ici, nous ne devons encore pas tenir compte des spécifications propres au matériel utilisé ! Ici, c'est en fait une retranscription du cahier des charges en des termes plus informatiques. Ainsi, on doit modéliser graphiquement les attentes du client, grâce à une méthode SA par exemple. En fait, on doit présenter toutes les options du programme vues de l'utilisateur avec les interactions qu'il y a entre elles. A ce stade du développement, nous devons également prévoir un Test de validation qui sera décrit ultérieurement.

2.4. Etude organique

Nous devons maintenant tenir compte des spécifications propres au matériel utilisé. Cette étape consiste à transcrire l'étude faite dans le dossier de spécification en la transposant du point de vue du programme. La méthode de représentation graphique peut être de plusieurs types. Personnellement, j'utilise les méthodes SA (pour le séquentiel) et Coad & Yourdon (pour l'orienté objet).

L'étude organique peut être divisée en plusieurs parties.

2.4.1. Etude graphique

Il faut tout d'abord transcrire les interactions entre les différents modules avec une "vue de haut". C'est à dire que l'on ne rentre pas dans le détail et qu'on ne présente que les modules principaux et les interactions entre eux. Si on utilise une méthode SA, on peut tout de même descendre d'un niveau ou deux.

On doit ensuite arriver à un détail des interactions entre les modules assez bas pour pouvoir écrire les algorithmes des différentes fonctions des modules. A ce stade, on doit prévoir les tests d'intégration.

2.4.2. Conception détaillée

Ici, on doit écrire les algorithmes des fonctions. Il y une règle d'or : l'algorithme d'une fonction doit tenir sur une feuille A4 (feuille de papier normale). Si vous avez des algorithmes plus longs, c'est que votre étude graphique est mal faite et qu'il faut diviser certaines fonctions en plusieurs fonctions.

A ce stade, on doit prévoir les tests unitaires.

2.5. Prototypage

Avant le codage à proprement parler, il faut effectuer des prototypages. Dans un prototypage, on teste la validité d'une solution proposée dans l'étude organique.

Les prototypages sont propres à l'environnement logiciel et matériel de votre logiciel. Ainsi vous pouvez faire un prototypage qui va tester le fonctionnement des timers sous Windows (je prends cet exemple car je l'ai fait il n'y a pas longtemps). Pour cet exemple, il est important de fixer une version de Windows. J'ai remarqué des différences importantes par exemple, dans la gestions des timers entre Windows 3.1 et la bêta version de Windows 95.

Dans les prototypages, on doit également créer une version minimum qui gère l'interface. Pour les applications sous Windows, c'est assez rapide mais nécessaire. Quelques fois, on peut remarquez des aberrations qui ont été faites dans les études précédentes.

2.6. Codage

C'est la phase la plus connue des informaticiens. Je rappelle qu'elle doit vous prendre au maximum 40% de votre temps de développement. Si vous suivez la méthode, tous vos algorithmes sont déjà créés et vous n'avez plus qu'à les retranscrire dans le langage que vous utilisez. Il est important aussi de noter que chaque listing doit être documenté. Pour chaque fonction, on doit indiquer :

```
- son nom,
- le traitement effectué,
- les entrées à fournir,
- les sorties qu'elle produit,
- la date de création,
- la date de mise à jour et
- la version.
```

2.7. Tests unitaires

Il doit exister un test par fonction.

La meilleur méthode pour les tests est de créer deux fiches pour chacun :

```
- un    plan de test qui est créé lors de la partie
    descendante du "Cycle en V" et
- une    fiche    de    test    qui est remplie lorsqu'on
    fait le test.
```

2.7.1. Plan de test

Cette fiche doit comprendre plusieurs rubriques :

- Entrée : Entrées qui sont fournies pour le test
- Sortie : Sorties qui sont attendues pour le test
- Traitement : Traitement qui doit être fait par le test
- Moyen de simulation : Si des entrées sont simulées (entrée du port série par exemple), il faut indiquer le nom du programme qui le simule.

2.7.2. Fiche de test

Cette fiche doit comprendre plusieurs rubriques :

- Entrée : Entrées fournies.
- Sorties constatées : Sorties qui ont été constatées lors du test.
- Validité du test : On indique si le test est valide. S'il ne l'est pas, on doit indiquer les problèmes constatés.

2.8. Tests d'intégration

Les tests d'intégrations sont fait sur plusieurs niveaux :

- l'intégration de certains modules entre eux,
- l'intégration de tout le programme.

La préparation et la réalisation des ces tests sont identiques aux tests unitaires. Il faut donc créer un plan de test lors de l'étude organique et remplir une fiche de test lors de la réalisation.

2.8.1. L'intégration de modules entre eux

Il faut vérifier que les interactions entre les différents modules sont valides. Il se peut que les modules fonctionnent bien indépendamment (ils ont été testés lors des tests unitaires) mais qu'ils ne fonctionnent pas ensemble.

2.8.2. L'intégration de tout le programme

Ici, on vérifie que le programme fonctionne dans son ensemble. Il s'agit en fait de vérifier qu'il n'y a pas de bugs techniques.

2.9. Test de validation

C'est le test le plus important du cycle. En effet, lorsqu'un client vous fournit un cahier des charges, vous devez créer un cahier de recette dans lequel figure un test réalisable sur le programme qui permet d'apprécier la validité du programme. Hum... Ca ne parait pas assez clair ?

Bon, imaginez que le client veuille que son programme ait une sauvegarde automatique toutes les dix minutes. Un des critères du test de validation sera :

- Une sauvegarde est réalisée au bout de 10 minutes.

Ce critère n'a rien à voir avec la partie technique. Bien sûr, il faut que le programme n'ait pas de bugs (ceci est vérifié lors des test d'intégration). Mais il se peut très bien que le programme n'ait pas de bugs mais qu'il ne soit pas valide.

Pour reprendre notre exemple, il se peut très bien que la sauvegarde fonctionne (testée lors des test organiques) mais que le délai de sauvegarde ne soit pas bon (testé dans le test de validation).

2.10. Les intéractions entres les phases du "Cycle en V".

Vous pouvez remarquer que le schéma comprend des flèches à double sens. En effet, si on s'aperçoit qu'il y a une fonction qui n'est pas réalisable dans l'étude organique, on adapte le dossier de spécification. Il en va de même pour la cahier des charges.

A ce propos, le cahier des charges qui est signé entre le demandeur et la personne qui doit créer le programme est toujours la version 1.1. Cette version comprend toujours des retouches faites par le programmeur. On trouve également des retouches de l'étude organique lorsqu'on se trouve dans le codage. En effet, les algorithmes comprennent souvent des fautes idiotes ou ne sont pas assez optimisées.

Les retouches sont aussi faites dans la conception détaillée lorsque les tests unitaires montrent des erreurs.

Ensuite il y a les retouches plus importantes et plus graves. Si vous avez un test d'intégration qui n'est pas valide, c'est à partir de l'étude organique de "haut niveau" qu'il faut faire des retouches, ce qui découle de nombreuses modifications dans la conception détaillée et donc dans le codage. Le pire qu'il puisse vous arriver ce sont des test de validation non valides, car là, c'est souvent toute la logique de votre programme qu'il faut remettre en cause.

3. Conclusion

Si vous avez eu le courage de lire tout cet article, vous avez sûrement pu voir les avantages de cette méthode. Je ne citerai que les principaux à mes yeux :

- on obtient exactement ce qu'on avait prévu,
- on sait exactement ce que l'on a dans notre listing et pourquoi on le possède,
- on peut s'y retrouver plus facilement par la suite,
- on dispose de nombreux documents qui permettent de savoir ce que fait exactement notre programme,
- la maintenance et l'évolution du programme peut être faite par une tierce personne.

Pas mal tout de même, non ? Ah oui... Si vous suivez cette méthode, vous remarquerez assez rapidement lors des études "graphiques" que des modules reviennent assez souvent, ou même des fonctions. Comme les fonctions sont documentées et succinctes, il est très facile de les réutiliser dans d'autres programmes. Il en va de même pour des modules entiers (surtout lorsque vous programmez en langage orienté objet).

Je n'ai pas mentionné toutes les documentations qu'il faut également réaliser (cahier de spécifications, cahier d'étude organique, cahier de prototypage, cahier de conception, cahier de tests, documentation, etc ...). Mais vous les ferez sûrement par vous même (là je rêve peut être un peu !). Toutefois, si vous comptez distribuer votre logiciel, il vous faudra automatiquement une documentation et un guide d'installation. Commencez ces guides dès le dossier de spécification ! Car si vous les réalisez à la fin, vous oublierez beaucoup de choses essentielles. Mettez régulièrement ces documentations à jour.

4. Et après ?

Si cet article vous a plu (écrivez moi, soit par lettre, soit par courrier électronique...) et si j'en ai le temps, je compte vous expliquer d'autres éléments de la méthode :

- méthode SA,
- méthode Coad & Yourdon et
- exemple complet.

Pour la méthode Coad & Yourdon, il faudra également faire un petit cours sur la programmation orientée objet. J'attends tous vos commentaires. Et j'espère que vous me ferez des programmes bien "propres" maintenant !

Tchernobyl, ça vous dit...?

JCR [Jul/Aou 95]

Tchernobyl ça vous dit quelque chose ?

L'expérience ayant conforté mon opinion et la grande confiance dans les services chargés de prévenir la population civile d'éventuelles pollutions radioactives, j'ai fais des frais et acheté un détecteur de radiations qui fait bip et allume une led a chaque radiation reçue. Modèle très simple pour ne pas dire simpliste.

Mon problème était de rentrer régulierement le nombre des clignotements pendant une durée (que j'ai choisie arbitrairement de 6 minutes) et à garder les valeurs relevées chaque jour.

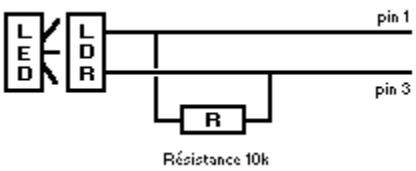
C'est faisable "à la main" mais pénible. J'ai donc voulu me servir de mon PC pour mesurer le temps et tester la led.

Ma solution :

J'utilise une LDR (résistance sensible a l'éclairement), ce n'est pas cher. j'ai pris une de 27 k (kilo-Ohms). Une résistance de 10 k en parallèle est mieux. Je rentre par la prise de jeux (joystick) les éclairs de la led du détecteur.

Schéma de principe :

Il faut s'arranger pour que la led soit presque contre la LDR, j'utilise une petite ventouse genre pistolet à flèche enfantin.



La liaison LDR -> prise, de plusieurs mètres est en fil souple
2 conducteurs, un blindage n'étant pas indispensable :

- pour le joystick 1, on peut mettre la LDR entre 1 et 3
  ou entre 1 et 6

- pour le joystick 2, on peut mettre la LDR entre 9 et 13
  ou 9 et 11

Dans le programme il faut évidemment tester la bonne entrée ! Attention, en Quick Basic, il faut toujours tester stick(0). Ce montage peut servir a tester ou compter n'importe quoi qui ait une à quatre leds montrant un état a surveiller !

Son GRAND AVANTAGE ?... AUCUNE LIAISON avec l'appareil, pas de risques pour le PC.

Je vous joins un petit programme d'exemple de lecture du joystick en basic, celui que j'utilise est en Turbo Basic (un vieux bidule de Borland) que je préfère. Et je stocke les mesures dans un fichier binaire que ce basic manipule très facilement, ainsi que les appels fonctions DOS.

NB : Notez dans le programme la boucle utilisée pour compter correctement le nombre d'éclairs de la led. Les valeurs lues par l'instruction stick(x) varient avec le montage et la vitesse du micro, aussi est-il nécessaire de l'ajuster pour que tout fonctionne bien.

Amicalement,

JCR

Lamerland

Warrant [Aou 95]

Un nouveau parc d'attractions est sur le point d'ouvrir.

Il est destiné aux Vrais Programmeurs, aspirants à quelques instants de détente entre deux oeuvres d'art en assembleur. En effet, le thème de ce parc est essentiellement axé sur la chasse aux lamers.
L'accès sera réservé aux possesseurs d'un permis de type 'ASM-Intermédiaire' ou mieux 'ASM-Avancé', obligatoire pour profiter du programme complet.
En tant que R.O.R, j'ai pu bénéficier d'un avant-goût des réjouissances qui seront très bientot offertes au public.
Bien que l'idée soit intéressante, mon avis reste néanmoins mitigé.
Je vous propose de vous narrer par le commencement cet étonnant safari...

* *
*

Lorsque mon guide m'enleva mon bandeau, je fus frappé par la beauté sauvage du site où l'on m'avait amené.
Mon guide s'empressa ensuite d'ôter les 3 mètres de cordages qui entravaient mes mouvements.

Je m'étirais, et me remis rapidement une vertèbre en place. Le long voyage en hélicoptère n'avait pas été de tout repos. Nous avions affronté des zones de fortes turbulences, et je sentais encore dans mon dos les stigmates laissés par un joystick Gravos, qui s'était malencontreusement retrouvé derrière moi.

Mes hôtes avaient tenu à garder notre destination secrète, et lorsque je m'étais présenté à l'aéroport, avec ma carte de R.O.R et mon invitation, l'on m'avait fait monter dans cet imposant hélicoptère Puma, avant de me placer un bandeau sur les yeux. Le voyage avait duré longtemps, mais nous étions maintenant arrivés.

Un désert à perte de vue.

Ici et là, des concrétions rocheuses formaient quelques timides affleurements sur une terre aride et plate.
Le vent soulevait des volutes de poussière, qui prenaient de curieuses teintes sous les feux du soleil mourant.
Le sol était crevassé par endroits, et une éolienne couinait faiblement derrière moi (très bon ça l'éolienne pour l'ambiance).

C'est alors que le Razorback apparut..Euh.. Attendez là... Hum, excusez moi. Je reprends.
C'est alors qu'un individu apparut. Il était de haute taille, et portait un treillis couleur sable. Il me dévisagea lentement, tandis que ses traits durs se morphaient en un sourire de bienvenue.

- Je vous souhaite la bienvenue à Lamerland. Je serais votre guide.

Il me tendit une carte.

- Ceci est une accréditation. Il me semble préférable de munir les visiteurs d'un pass, surtout lorsqu'ils sont difficiles a distinguer d'avec le gibier.

"Ben merde alors !" m'exclamai-je en mon for intérieur. En effet,il y a quelque chose de vexant à être assimilé à la triste tribu que sont les lamers, surtout lorsqu'on cultive sa ressemblance avec les programmeurs-authentiques-et-rebelles. (J'envisage d'ailleurs même de me mettre à programmer.) Je lui signalais quand-même que loin d'être un lamer, j'était un de ces autodidactes qui se sont faits tout seul (avec les sources des autres).

Il reprit la parole:

- Au fait,vous avez peut-être entendu parler de moi..On me surnomme Javel-Net.

Si j'avais entendu parler de lui ! Javel-Net, le nettoyeur des réseaux ! Figure quasi-mythique, chimère pour certains, maître spirituel pour d'autres. L'on prétendait qu'il sentait les lamers, à peine ceux-ci connectés. Aujourd'hui encore, je ne doute pas un seul instant qu'il ait usurpé cette identité. C'était lui, j'en reste persuadé. A en juger pas le flou au delà de l' éclat métallique de ses yeux, il était évident qu'il avait passé son enfance dans un placard à balais, enchaîné à un minitel.

Un second individu nous rejoignit. Mon guide fit les présentations. Il se nommait Karl Ouane, et était un ancien sysop chargé maintenant de la sécurité. Karl me jeta un coup d'oeil soupconneux, et me tendit un bloc de parpaings. Apres quelques instants, je réalisais qu'il ne s'agissait pas de parpaings, mais bien de ses phalanges, et qu'il se proposait de me serrer la main. Les douleurs se sont un peu atténuées aujourd'hui, et je pense que je pourrais bientot enlever mon bandage.

Karl nous quitta, tandis que je montais dans un monstrueux TerraFucker, dont l'avant s'ornait d'un magnifique pare-buffles. Javel-Net m'expliqua que nous nous rendions vers un petit point d'eau, afin de commencer notre chasse. Cela me paru logique, car ou trouver des lamers, sinon près d'une source ?

Une fois arrivés, nous nous embusquâmes et attendîmes. Le crépuscule s'installait doucement, et nous ne vîmes qu'au dernier moment une forme furtive qui s'approchait. Il s'agissait d'un adolescent boutonneux, tenant sous son bras le dernier numero du 'Monde de Lamer'. Javel-Net me tendit un fusil à pompe, en me chuchotant "c'est un branché ! Pas de pitié avec cette engeance !" Le branché entendit le claquement sec du fusil que j'armais, et se tourna vers notre cache. En nous apercevant, il dit tout d'abord :

- Hey les mecs ! J'suis un Cybernaute Cyberpunk. Vous auriez pas vu le Cyberspace ?

Puis en apercevant mon fusil, il ajouta :

- Oh p'tain mec ! T'es Cybergrave ! Cyberdéconne pas !

Ce furent ses dernières paroles. La chasse commençait bien.

Nous attendîmes ensuite, mais personne ne vint, les détonations avaient sans doute alerté les lamers à proximité. La nuit était maintenant bien là. Il nous fallait établir notre campement. Javel ouvrit une fenêtre du 4x4, et tira-copia un lourd sac contenant tous les ustensiles nécessaires aux aventuriers de l'extrême de notre époque (antenne satellite,balise Argos,télevision 16/9 ,modem,fax,...).

Karl nous rejoint avec le reste de l'équipement. Tout en devisant gaiement autour d'un feu de camp, nous restions attentifs aux bruits alentours, espérant une dernière proie avant les heures profondes de la nuit, ou même les lamers n'osent plus ripper.

Notre attente ne fut pas déçue. Un bruit de pas se rapprochait, et un individu de haute taille en costard cravate, et portant un attaché-case pénétra dans le rayon de lumière projeté par notre brasier.

- Messieurs, bonjour. Je dois participer a une conférence sur le Multimédia, mais me suis égaré dans cette triste contrée. Auriez vous l'extrême amabilité de me reconduire dans des lieux plus civilisés ?

Il s'épousseta du revers de la main, attendant notre réponse. Je lui demandai :
- Sur quoi porte la conférence ?

Il me repondit d'un air agacé :
- Mais sur le Multimédia,sous toutes ses formes. Le formidable potentiel de communication qui va réunir les peuples au sein du village planétaire, et qui fera disparaître nos modes de contacts primitifs.

Karl sourit de toutes ses dents, spectacle par ailleurs impressionnant.
- Des modes de contact primitifs ,avez-vous dit ? Que pensez-vous de celui-ci ?

Et sans lui donner le temps de répondre, son bras opéra un large moulinet dans sa direction. A l'extrémité de son bras se trouvait une grosse hache qu'il avait à côté de lui depuis le début du repas. Le costard-cravate ne semblait pas très ouvert, mais je fus bien obligé de changer d'avis en voyant la crevasse béante qui ornait maintenant le sommet de son crâne. Il s'écroula en gargouillant "interactivité" et "convivialité". Karl se frotta les mains d'un air satisfait et alla déchirer un annuaire pour se détendre. Javel lanca a Karl :

- Karl, tu veux que je te disk ? Tu prends ton boulot trop stacker !

- Je sais, il faudrait que je décompresse...

Il était maintenant temps de dormir. J'étouffai un baillement, et posai une dernière question au nettoyeur des réseaux

- Vous avez des appats particuliers pour attirer le lamer ?

- Habituellement, nous utilisons un disk du 'Reporter', que nous laissons bien en évidence au milieu de la piste. Les lamers le sentent dans un rayon de deux kilomètres.

- Et ca marche à tous les coups ?

- Eh bien, parfois, le lamer s'enfuit avec le disk. Nous avons bien essayé un piège plus subtil, en distribuant le Reporter, et en proposant aux lamers des explications pour les articles qu'ils ne comprendraient pas, mais ils ne sont pas assez stupides pour tomber dans le panneau. Les seuls retours que nous avons eu émanaient de personnes se situant a l'opposé du lamer. Certaines ont d'ailleurs rejoint notre équipe, et travaillent en collaboration avec notre équipe de rabatteurs.

J'écarquillais les yeux, plissais le front, pinçais la bouche, haussais un sourcil et me grattais le crâne. Hélas, ces mimiques pourtant très expressives, et qui font d'habitude raconter leur vie aux personnes interviewées, ne me permirent pas d'obtenir davantage de précisions quant aux rabatteurs.

Il était donc grand temps de dormir, mais j'était plutot énervé.

Je me mis a compter le 'lamer-qui-saute-la-barriere-pour-piquer-les-sources', et je devais en être à plusieurs milliers lorsque Morphée daigna enfin appuyer sur OFF.

Je passais une nuit passablement agitée, peuplée d'archives corrompues et de bugs systemes.

L'odeur du café me réveilla. Le ciel se parait des couleurs douces de l' aurore, et les premiers oiseaux se raclaient la gorge.

Je me mis sur mon séant, cherchant des yeux mes compagnons. Karl sifflotait en buvant un liquide noir, qui à en juger par l'odeur qui m'avait reveillée, devait être le descendant direct du café que Jesus avait préparé pour Lazare. Je m'abstins donc prudemment, et avisais un lecteur de disquettes d'où sortait une bonne odeur de grillé. En m'avançant pour prendre un toast, je vis avec satisfaction que c'était un lecteur Amiga (c'est avec ces lecteurs que l'on fait les meilleurs grille-pain). J'entendis alors la voix de Javel-Net derrière moi :

- Ah, vous êtes réveillé ! Prenez donc un toast et du café. Au début, nous utilisions des grille-pain plus traditionnels, mais ceux-ci avaient une facheuse tendance à s'envoler au bout d'un certain temps de non-utilisation.

Ne cherchant pas à comprendre, j'engouffrais un toast 3"1/2, ce qui calma quelque peu les gargouillements indésirables qui commençaient à émaner de ma personne. Pendant ce temps, Javel me donnait le programme de la journée :

- Nous allons faire un tour dans un coin qui pullule de lamers. Nous l' appelons entre nous 'Lamer-Rock'. Il semble que ce soit un coin qu'ils affectionnent tout
particulièrement. Ensuite, nous vous reconduirons chez vous.

Karl avait finit de charger le 4/4. Il ferma quelques fenêtres pour récupérer un peu de mémoire et mit le contact. Le 4/4 émit un bip et refusa de démarrer. Karl sourit d'un air légèrement géné :

- Heu, c'est un petit problème dans le moteur. Ne vous inquiétez pas, je vais le patcher avec ce trombone.

Ce qu'il fit, et nous pûmes enfin partir. Lorsque Javel annonça notre destination, Karl lacha un instant le volant, et joignit les battoirs qui lui servaient de mains.

- Oh c'est vrai ?! Je pourrais faire un carton avec mon Bazou-K ?

- Voyons Karl, ne sois pas si goinfre ! Il faut bien qu'il reste quelques lamers pour nos visiteurs ! , répondit Javel

Karl se renfrogna

- Oh bon, si on peux pas rigoler de temps en temps...

Javel sortit son portable, et l'alluma pour mettre en route son scanner de lamers.

- Vous devriez jeter un coup d'oeil dans le répertoire "C" , lui suggérai-je

- Pourquoi ?

- Parce que "Lamer qu'on voit dans C" (ND.Warrant: Arfarf, promis,j'arrête!)

Le scanner bippait avec insistance. Karl coupa le moteur, et nous continuâmes a pied. D'énormes rochers masquaient le paysage, et nous dûmes les contourner.

Nous arrivâmes alors face a un spectacle étonnant. De gigantesques pancartes annoncaient : "Forum Multimédia". Un bruissement de voix nous parvenait, avec ça et là quelques mots identifiables : "marchés", "CA", "KF" et autres joyeusetés.

L'entrée était bien entendue payante, mais Javel sortit des badges qu'il nous tendit. Il n'était bien sûr pas question de décliner mon identité de ROR, ce qui aurait semé la panique parmi le gibier.

Nous accrochâmes nos badges, sur lesquels etait inscrit: "Pc-Multimedia- Internet-Saucisse-Rillettes".

Nous passâmes ainsi sans problèmes l'entrée. Karl était devenu blême d' excitation, et salivait presque en contemplant la pléthore de lamers qui circulaient dans les stands. Je fus très impressionné par cette profusion, et me demandais un instant comment avaient-ils pu organiser cette foire en plein milieu du parc d' attraction.

Ignorant les conférences alléchantes du style "Comment placer en toute occasion «multimédia»" , nous nous frayâmes un chemin au milieu d'une cohorte de pigistes émerveillés qui regardaient les yeux grands ouverts un individu leur faire une démonstration d' "une séquence multimédia interactive conviviale" (Une animation pornographique saccadée)

Nous nous dirigeames vers le stand "Oncenfoudukod", ou nous accueillit Mr Pourvukonvende.
Il arrive que le gibier inspire pitié. Le responsable de ce stand n'était manifestement plus responsable de ses actes. Il semblait avoir subit le conditionnement "Widow". Il nous débita d'une voix monocorde une longue tirade sur la gestion des ressources, et nous promis une version de "Widow" qui résoudrait les problèmes de compatibilité avec elle-même.
Je lui demandais s'il connaissait "Warf", et il me dévisagea sans comprendre, tel une personnalisation de l'incompréhension. Sans qu'un mot ne soit échangé, Karl tira son opinel Gonzo, et lui trancha la gorge.

Avec le sentiment d'avoir fait une bonne action, nous quittâmes le stand, et de là, le forum.
Il était maintenant temps pour moi de regagner l'hélico qui me ramènerait à mon paisible foyer.

Sur le chemin du retour, pas un mot ne fut échangé. Javel me regardait en coin, d'un air que je n'appréciait pas trop. Au moment de monter dans l'hélico, il me dit finalement :

- Décidemment, vous avez de plus en plus une tête de lamer.

Je vis sa main s'approcher dangereusement de la poche intérieure de sa veste. Ma vie défila dans ma tete en 0,99978 secondes (Pentium inside).

Il hésita, et sa main ressortit de la poche sans objet susceptible d' occasionner des dommages, du moins sur ma personne, car je ne pourrais pas en dire autant pour sa part. En fait, il venait de sortir un paquet de cigarettes.

Je compris qu'il était temps de nous séparer, et me hâtai de monter dans l'hélico.
Le pilote me banda a nouveau les yeux, et me ficela en sifflotant. Nous regagnâmes d'une traite mon lieu de départ. J'ai l'impression que nous avons passé un certain temps à tourner en rond, mais je n'en jurerais pas.

* *
*

Je viens d'apprendre que Karl et Javel-Net viennent d'échapper à un attentat revendiqué par la FAC (Fraction des Authentiques Codeurs).
Un tract est passé sur les réseaux, dans lequel on démontre avec force détails qu'ils ne sont que de pauvres lamers.

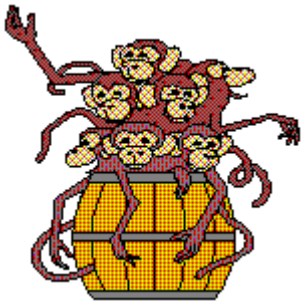
Quand à moi, j'ai reçu ce matin une invitation pour l' Uranus Party 51. Je me demande si je vais y aller. Le rendez-vous pour le bus est fixé mardi prochain a minuit, au quatrième sous-sol de la fonderie désaffectée.

Contact : RTEL / TEASER (Bal: Warrant)

Et Ping ! pour Unisys...

François CLAUSTRES (Léviathan) [Jun 95]

Zipper... en images !



Dans notre précédent numéro, nous avions évoqué quels étaient les meilleurs compresseurs "Sourciciels" (sourceware) : GZIP et Info-ZIP, tous deux basés sur l'algorithme "DeflatX" de Phil Katz ZIP (PKZIP), que celui-ci a mis dans le domaine public.

Cette fois-ci, nous allons parler du successeur désigné de GIF (Graphic Interchange Format) : le format PNG (Portable Network Graphics) - à prononcer "ping" pour ne pas passer pour ringuard SVP ! En effet, pour de sombres raisons de brevets détenus par Unisys sur l'algorithme de compression "LZW", les utilisateurs du très répandu format GIF (programmes de dessin, serveurs BBS ou Internet...) doivent payer une licence pour avoir le droit de l'utiliser...

Or aucun autre format existant ne pouvait prendre la relève du GIF, dont la particularité est d'une part de compresser très efficacement **SANS PERTES de qualité** (au contraire de JPEG), et d'autre part d'offrir un format d'affichage "entrelacé" qui permet d'avoir une vision globale de l'image, avant que celle-ci soit complètement chargée (particularité très appréciée sur Internet/WEB!).

Aucun rapport entre cette fois-ci et la précédente, pensiez-vous ? Que nenni chers amis ! Car figurez-vous que l'algorithme de compression utilisé dans PNG est précisément le "**DeflatX**" de Phil Katz... Choisi naturellement, parce que l'un des plus répandu (y compris côté sources !), des plus fiables... Et des plus "domaine public" !

Au fait, pourquoi ne pas avoir créé un format GIF avec un algorithme "DeflatX" ? Après avoir étudié cette hypothèse, les auteurs du format PNG (parmi lesquelles on retrouve le fameux J.L. GAILLY !) ont jugé que c'était une fausse bonne idée : la plupart des viewers actuels ne le reconnaîtraient pas (ceux qui ont bataillé avec les "standards" TIFF savent de quoi je parle !), et en plus créer un nouveau format n'est pas tellement plus compliqué, vu les nombreuses modifications que l'implantation d'un "DeflatX" dans GIF imposait !

Et accessoirement, cela permettait de rajouter quelques améliorations, comme le mode True Color qui n'existait pas dans GIF (limité à 256 couleurs).

Après avoir étudié rapidement les spécifications de ce format, nous verrons quelles sont ses performances côté taux de compression et vitesse de décompression.

1) Que fait PNG ?

Puisque PNG a été créé pour remplacer GIF, le plus simple est encore de procéder par comparaison avec ce dernier (rapide résumé de ce que j'ai lu dans les spécifications officielles de PNG) :

1.1) Ce qu'il a repris à GIF :

- Images bitmaps avec palette, de 2 à 256 couleurs ;
- Mode entrelacé spécialement dédié aux communications ;
- Possibilité de transparence (pour de jolies mises en pages) ;
- Ajout de textes pour commenter chaque image ;
- Indépendance vis-à-vis de quelque système que ce soit ;
- Compression sans perte de qualité (i.e. sans modification du contenu de l'image).

1.2) Ce qu'il rajoute/améliore par rapport à GIF

- Images "truecolor", jusqu'à 48 bits par point ;
- Images "niveaux de gris", jusqu'à 16 bits par point ;
- Gestion des masques de transparence ("full alpha channel"), que ce soit en mode truecolor ou en niveaux de gris ;

- Inclusion d'un indicateur Gamma de luminosité, pour que le viewer affiche correctement l'image sans interventions (mais l'image en elle-même n'est pas modifiée par ce Gamma) ;

- Détection des erreurs (CRC) améliorée ;
- En mode entrelacé, la première vision globale de l'image (avec de gros carrés) arrive HUIT fois plus vite qu'avec GIF ;

- Pas de problèmes de brevets pour quelque partie du format que ce soit.

1.3) Ce qu'il n'a pas (encore) repris à GIF

- Le format GIF89, qui permet de sauver plusieurs images dans un même fichier ; mais c'est prévu, à terme.

2) Quelles sont ses performances ?

2.1) Coté compression : le meilleur, haut la main !

Deux exemples valent mieux qu'un long discours... J'ai choisi d'une part une image noir&blanc (tirée de la numérisation d'un tract électoral...), car les performances de GIF étaient quasi-imbattables sur ce genre de format (que JPEG ne traite pas par nature, de toute façon !) ; et d'autre part, une image "ray-trace" 640x480 en 256 très difficile à compresser (compte tenu du nombre de nuances).

Résultats résumés dans ce tableau (la première ligne donne la taille en octet, la seconde le pourcentage par rapport au fichier BMP) :

	BMP		PCX	TIF	GIF	PNG
	(non compressé)		(PakB)	(non-entr.)	(non-entr.)	
827x1169 2c	121638	81758	57914	42134	35259	
% du BMP	100%	62,21%	47,61%		34,64%	**28,99%**
640x480 256c	308278	364669	302526	222608	198697	
% du BMP	100%	118,29%	98,13%	72,21%	**64,45%**	

Note : ces essais ont été réalisés grâce à COMPUSHOW 2000, un des premiers shareware qui, dans sa version 2.0a, a adopté le standard PNG.

Globalement, PNG compresse mieux que GIF (un gain de 10 à 20%)... Cela est assez logique, dans la mesure où "DeflatX" est toujours plus performant que "LZW" ! (Comme quoi nous devrons remercier Unisys de leur initiative : sans eux, PNG n'aurait jamais vu le jour !)

2.2) Vitesse de décompression & d'affichage : peut mieux faire !

Il y a deux façon de voir le problème en fait :

- En *mode "entrelacé"* : c'est le point de vue BBS/Internet, où la "vision" approximative - mais globale - d'une image importe. En effet, on peut par exemple cliquer sur un bouton-image avant que celui-ci soit complètement affiché dans tous ses détails pour passer à la suite, du moment que l'on voit grosso-modo de quoi il s'agit !

Or PNG est beaucoup plus rapide que GIF à afficher une vision "floue" mais globale de l'image : pour être exact, **8 fois plus rapide que GIF**, pour une première version intégrale (avec de gros carrés) de l'image... De plus, il intègre des tests d'intégrité plus performants que ceux de GIF pour faire face à des perturbations "structurelles" d'Internet (qui peut confondre une image et du texte) ; mais là je n'ai pas pu tester "de visu".

Donc : mention **TRES BIEN** pour cet usage.

- En *mode "non-entrelacé"* : c'est le point de vue de celui qui utilise ce format pour stocker des images dans un espace minimum, et sans pertes.

Là, il faut déchanter ! Autant l'affichage de GIF peut être considéré comme rapide, est sans à-coups avec la plupart des viewers, autant les images PNG apparaissent lentes à s'afficher, et le font de façon saccadée.

Impression confirmée par le chronomètre : l'image "ray-trace" sus- citée s'affiche avec COMPUSHOW 2000 en moins de 3 secondes quand elle est au format GIF, et en un peu moins de 9 secondes quand elle est au format PNG. Soit **TROIS fois plus lentement**. (Tests sur 486sx33 VLB, carte Cirrus Logic VESA)

Ce n'est pas grave si la fonction essentielle de PNG est le stockage, mais c'est plus ennuyeux si on veut l'utiliser pour faire des "slide-show"...

Il faut espérer que comme pour JPEG, les programmeurs fassent un effort pour compenser cette (relative) lenteur... Le format ZIP étant pourtant un des plus rapides (si ce n'est le plus rapide de sa catégorie !) à la décompression, la marge dont ils disposent apparaît toutefois limitée... Mais bon, on peut toujours espérer !

Autre critique : les auteurs de PNG ont tenu à incorporer la correction Gamma **DANS** le format de l'image... Or cela va à l'encontre de l'objectif initial de "faire simple" pour l'utilisateur, car il faut avoir lu la documentation officielle de PNG pour savoir (deviner...) comment régler ce paramètre (à 0,45 ou 450/1000) lors de la sauvegarde, afin que la luminosité soit correcte lors de l'affichage du fichier PNG... (Mais Compushow aurait pu aussi être plus clair dans son aide à ce sujet !)

Donc : mention **ASSEZ BIEN** seulement pour cet usage.

Au final, il est clair que je passerai au format PNG dès que celui- ci sera largement supporté par les programmes graphiques; en particulier pour le format noir&blanc ou le 16 couleurs, où les défauts sus-cités (relative lenteur) sont beaucoup moins apparents.

Léviathan.

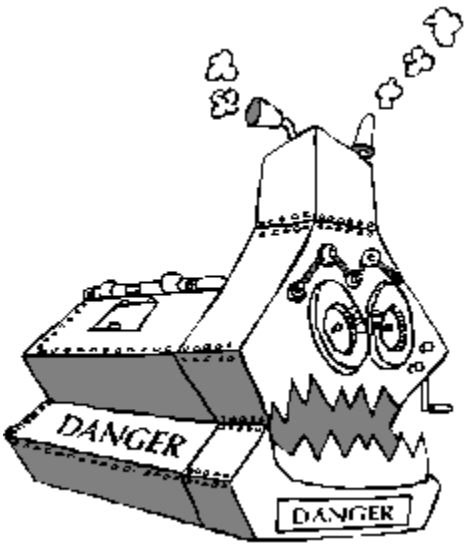
Pour en savoir plus...

Les "PNG (Portable Network Graphics) Specifcation" (la mienne : 9e ed., du 7 mars 1995), sont disponibles sur :

- sur 3614 TEASER : **PNG9WW.ZIP** (fichier au format WinWord 6)
- sur le WEB : **http://sunsite.une.edu/boutell/png.html**

Mathématiques et Grands Nombres

Scythale [Jun 95]



Tous ceux qui, un jour, ont eu à rendre un programme, ou pire à faire pour leur propre satisfaction, un programme utilisant des fonctions mathématiques utilisant de grands nombres, se sont peut-être heurtés au problème des limitations de type du Turbo Pascal. Grande nouveauté avec le BP7, les formats COMP, mais les nombres sont encore très limités. Suite à un projet de recherche de grands nombres premiers, qui sont tous positifs, car je ne parlerai que de grands nombres positifs, la première chose que mon binôme à faite, a été d'utiliser les listes, dont il avait l'habitude. Mais tout ceux qui sont habitués à utiliser le Pascal sur des stations VAX/VMS se retrouvent marris devant leurs pauvres PC quand ils s'aperçoivent de la terrible limitation de leur système en ressource dynamique : le Pascal ne désallouant pas après un appel (j'envoie 2 listes, et j'en renvoie une dans la 1e i.e. NA := add(NA,NB), l'ancien NA reste en mémoire.

Dans ce cas-là, que faire ? Passer en mode protégé est une solution, mais quand on utilise des routines asm, tout recalculer c'est la poisse..., ou tout bêtement créer de nouvelles fonctions de grands nombre. C'est donc cette méthode que j'ai choisie pour contourner cet écueil, et dont la seule limitation était la mémoire dynamique du système (on peut toujours y appliquer le mode protégé). L'idée est simple, deux tableaux en l'occurrence pour une suite de fibonnacci, tous deux de 32000 cases de byte, puis créer de nouvelles fonctions d'addition, de soustraction ... Avantage immédiat : une case de liste fait 5 byte, une de tableau 1, d'où un gain conséquent de mémoire.

Le type est défini juste ici :

```
type
numtab = array[1..32000] of byte;

BIGnum = record
t : ^numtab;
s : integer;
end;
```

Le record Bignum a deux champs, le premier T le pointeur sur tableau de byte, et le deuxième, la taille du nombre contenu dans T, très utile pour optimiser le calcul.

Pour les utiliser, on les crée par :

```
var NA : BIGnum; (* on les définit *)

begin
new(NA.T); (crée ce nombre, A NE JAMAIS OUBLIER SOUS PEINE DE MORT)
init_BN(NA);      (on le met a 0)

add_c(NA,2);      (on met 2 dans NA, c.f. plus loin...)

release(NA.T); (détruit NA)
end;
```

Voyons les fonctions mathématiques de base :

```
     {------[ Procédures mathématiques ]-------------------------17/05/95-}
```

On initialise en mettant des zéro partout et en mettant la taille à 0 :

```
procedure init_BN(var ptr:BIGnum);

(Initialise un grand nombre à zéro)
var i : integer;
begin
for i := 1 to 32000 do
ptr.t^ [i] := 0;
ptr.s := 0;
end;
```

Renvoyer le plus grand est souvent nécessaire pour deux Integers :

```
                function max(i,j:integer):integer;

(Renvoie le plus grand des Integers)
                begin
                if i>j then
                max := i
                else
                max := j;
                end;
```

Il faut déjà implémenter une fonction pour savoir si on a affaire à un nombre nul, on ne sait jamais, mais ça va servir, je vous le promets :

```
                function est_nul(NA:BIGnum):boolean;
                begin
                est_nul := (NA.s=0) OR ( (NA.s=1) AND (NA.t^[1]=0) )
                end;
```

Première procédure intéressante après l'initialisation, l'addition de base, c'est à dire celle de plus bas niveau, l'addition par une constante inférieure à 10. (C.f. cours de numérologie). J'appelle par la suite BN un Big Number, un grand nombre.

```
                procedure add_c(var NA:BIGnum;cst:byte);
                var ret : byte;                       (la retenue)
                i : integer;                             (le compteur)
                c : integer;                             (le calcul intermédiaire)
                Begin

                ret := 0;                                    (pas de retenue au départ)

                if not((cst=0) or est_nul(NA)) then                  (zéros, rien a
traiter)
                for i := 1 to na.s+1 do              (on va de 1 a n+1, en cas de
retenue)
                begin

                if i>1 then cst := 0;                          (idiot, mais utile pour
optimiser)

                c := NA.t^[i] + (cst + ret);                    (valeur actuelle plus
cst si)

(premier rang, et retenue si)

(il y a lieu)
                ret := 0;                                                    (on
l'a utilisée, on la retire)

                if (c>=10) then                              (si il y a retenue dans
notre calcul)
                begin
                ret := 1;                                                 (on la met
pour le prochain tour)
                c := c-10;                                               (on retire
10 de c)
                end;

                NA.t^[i] := c;                  (on met dans la position du BN, la
valeur c)

                if (ret=0) then                                      (plus de
retenue, on peut sortir)
                exit;
(allez, hop on sort, c'est fini)

                end;                                                          (fin
du FOR BEGIN)

                if (NA.t^[na.s+1]<>0) then inc(NA.s);                    (si le BN
a grandi)

                end;
```

Et maintenant l'addition complète :

```
                procedure add(var NA, NB, ND : BIGnum);
                var ret : byte;
                i : integer;
                c : integer;
                Begin
                ret := 0;                                   (pas de retenue)

                if not(est_nul(NB) or est_nul(NA)) then                         (dès
0, on sort)
                begin
                for i := 1 to max(NA.s,NB.s)+1 do               (on calcule au plus
grand)

(des deux+1 si retenue)
                begin
                c := NA.t^[i] + NB.t^[i] + ret;
                ret := 0;
                if c>=10 then
                begin
                ret := 1;
                c := c - 10;
                end;
                ND.t^[i] := c;
                end;
                if (ND.t^[i]<>0) then ND.s := i                 (si il y a du monde
en n+1)
                else if (i>1)and(ND.t^[i-1]<>0) then ND.s := i - 1;
(sinon)
                end
                else
                    ND:=NA;
                end;
```

Rien de bien nouveau et pas d'astuces, mais voyons la soustraction d'une constante :

```
                procedure sous_c(var NA : BIGnum;cst : byte);
                var ret : byte;
                i : integer;
                c : integer;
                Begin

                ret := 0;
(pas de retenue)

                if not est_nul(NA)and(cst<>0) then        (si NA est nul, rien a
faire)
                for i := 1 to NA.s do

                begin

                if i>1 then cst := 0;
(c.f. add_c)

                if (NA.t^[i]<(cst+ret)) and (i<na.s) then

                begin                                              (si on retire plus
qu'il n'y a, retenue)

                NA.t^[i] := 10 + NA.t^[i] - ( cst + ret );
                ret := 1;

                end
                else
                begin                                    (sinon simple soustraction, et
fin de calcul)
                                                              (car plus rien a retirer)
                NA.t^[i] := NA.t^[i] - ( ret + cst );
                ret := 0;

                if (na.s=i) then
(dernière case)
                if (i>0) and (NA.t^[i]=0) then
                dec(NA.s);
(si pas dernière case)

                exit;

                end
                end;

                if (NA.t^[i]<>0) then NA.s := i                                (si
taille inchangée)
                else if (i>1) and (NA.t^[i-1]<>0) then NA.s := i-1;
(sinon)

                end;
```

Ici une optimisation, renvoie le plus grand, comme on travaille seulement en nombres positifs, donc on soustrait le plus grand au plus petit :

```
                function greater(NA,NB:BIGnum):boolean;
                var ret : byte;
                i : integer;
                c : integer;
                Begin

                if NA.s>NB.s then                              (on commence par
regarder la taille)
                greater:=true
                else if NA.s<NB.s then
                greater:=false
                else                                                          (et
après comparaison case a case)
                begin
                i:=max(na.s,nb.s);
(taille max)
                while (NA.t^[i]=NB.t^[i])and(i>0) do                 (boucle tant
qu'égaux)
                dec(i);                                              (sens
décroissant, c'est plus rapide)
                if (NA.t^[i]>=NB.t^[i]) then
                greater:=true
                else
                greater:=false;
                end;

                end;
```

Et maintenant la soustraction, sous vos applaudissements :

```
                procedure sous(var NA,NB,ND:BIGnum);
                var ret : byte;
                i : integer;
                c : integer;
                Begin
                ret := 0;
                if est_nul(NB) then
                ND:=NA
                else
                for i := 1 to max(na.s,nb.s) do
(le parcours)
```

```
                begin

                if (NA.t^[i]< (NB.t^[i]+ret) ) then
(A<B+retenue)
                begin

                ND.t^[i] := 10 + NA.t^[i] - ( NB.t^[i] + ret );
                ret := 1;

                end
                else
(A>B+retenue)
                begin

                ND.t^[i] := NA.t^[i] - ( NB.t^[i] + ret );
                ret := 0;

                end;

                end;

                if (ND.t^[i]<>0) then ND.s := i
                else if (i>1) and (ND.t^[i-1]<>0) then ND.s := i - 1;
                end;
```

Voyons maintenant d'autres fonctions plus diverses, par exemple pour savoir si il y a reste dans la division de A par B :

```
                function modulozero(var NA,NB:BIGnum):boolean;
                var naa : BIGnum;
                Begin
                new(NAa.t);
(on crée une nouvelle)
                init_BN(NAa);
(on l'initialise)
                add(NA,NAa,NAa);
(on met NA=NAa)

                while greater(NAa,NB) do            (puis on soustrait NB a Naa tant
que)
                sous(NAa,NB,NAa);                          (Naa>NB)

                modulozero:= not (is_void(NAa));

                dispose(NAa.t);                                                  (on
nettoie avant de quitter)
                end;
```

Et maintenant la division par 2, mon cours de numérologie me dit que diviser un nombre par 2, c'est trouver le nombre de 2 dans celui-ci, je vais donc soustraire A div 2 fois 2 pour trouver A div 2 :

```
                procedure div2(var NAA:BIGnum);
                var t : BIGnum;
                m : longint;
                i : integer;
                Begin

                m := 0;
                if (naa.s<=10) and (naa.t^[10]<=2) then
                begin            (taille < 10, on peut utiliser des longint, compat.
tp6)

                for i := naa.s downto 1 do
                m := m*10 + naa.t^[i];                                    (on passe
de BN a longint)
                m := m div 2;
(on divise par 2)

                init_BN(naa);
(on passe de longint a BN)
                i := 0;
                while (m<>0) do
                begin
                inc(i);
                naa.t^[i] := m mod 10;
(on prend le dernier)
                m := m div 10;
(et on le retire)
                end;
                naa.s := i;
(on ajuste la taille)
                end
                else                                    (on est condamné à des
calculs très longs...)
                begin
                new(t.t);
                init_BN(t);

                repeat
                sous_c(NAA,2);
(on retire 2)
                add_c(t,1);
(et on le comptabilise)
                until (NAA.s<=2) and (NAA.t^[1]<=1) and (NAA.t^[2]=0);

                init_BN(NAA);
                add(NAA,t,NAA);                                      (on renvoie t
qui est naa div 2)

                dispose(t.t);
                end;

                end;
```

Et après ca, il faut les afficher ces fichus nombres :

```
                procedure show(ptr:BIGnum);
                var i : integer;
                s : string;
                begin

                for i := ptr.s downto 1 do
                begin
                str(ptr.t^[i],s);
                affiche(s);               (on affiche un digit, une fonction de
votre cru)
                end;

                end;
```

Limitations :

Evidemment, on ne travaille qu'avec des nombres positifs, mais dans le record BIGnum, il est très facile de rajouter un booléen de signe. La mémoire, chaque tableau de 32000 cases fait ... 32 Ko au moins, penser à allouer une Heap conséquente. {$M , , } ...

Il y a des bugs, ce n'est pas optimisé... je vous vois venir, le but de cet article est de présenter des méthodes de grands nombres, si vous avez mieux, ca m'intéresse, et j'attends vos critiques.

Conclusion :

Vous voici tout clef en main, vous n'avez plus qu'à faire du copier-coller ... L'exemple est le programme FIB.EXE.

Bibliographie :

Dr Dobb's #226 janvier 95, avec un article sur les grands nombres en C++.

Principes fondamentaux de l'informatique, Ullman ed.DUNOD que je vous recommande vivement.

Remerciements :

Mon prof A. Souah pour ses supers cours, bien que j'attends toujours la dernière éval et mon cours de numération,

HP Mâd, pour son cours d'assembleur Flash qui me « sert » de modèle,

Murdoc, mon binôme, qui malgré ses défauts n'en est pas moins sympa,

et Ecureuil pour ses conseils, et ses routines sans qui certains de mes programmes ne seraient pas ce qu'ils sont.

Et toi, hypocrite lecteur, mon ami, mon frère. (Baudelaire)

Grands Nombres et Listes

Scythale [Emmanuel Pierre] Jun 95

Suite à l'article où je décrivais les grands nombres à travers les tableaux, voilà pour tous ceux qui sont obligés d'utiliser les listes, les fonctions correspondantes.

La grande différence avec les tableaux, est que les tableaux se traitent de manière linéaire, alors que les listes se traitent de manière récursive.

Qu'est-ce que la récursivité ?

C'est la propriété qu'a une fonction de s'auto-appeler, c'est à dire de se rappeler elle-même plusieurs fois. On décrit ça en disant que l'on ouvre plusieurs feuilles de calcul en simultané.

Pratiquement, une fonction qui se rappelle stocke sur la pile ses données, et son point de retour à la fin de la fonction appelée sur le stack, et quand la sous-routine se termine, les paramètres de la précédente (appelée mère), sont restaurés.

Théoriquement, il y a deux types de récursivité, la récursivité terminale, et la récursivité non-terminale.

La récursivité non-terminale, c'est quand on rappelle la fonction avec les nouveaux paramètres, et qui se termine en renvoyant le résultat, c'est à dire que le résultat va remonter toutes les feuilles de calcul.

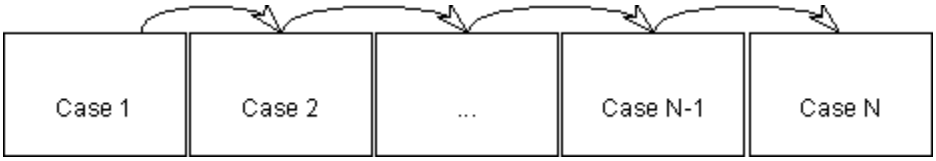
Pour la récursivité terminale, on rappelle la fonction en lui passant un accumulateur, c'est lui qui contiendra le retour, et évitera ainsi de tout désempiler lors du retour.

Les deux méthodes sont équivalentes en temps machine, mais il faut les connaître, c'est très utile suivant le type de calculs à effectuer.

Qu'est-ce qu'une liste ?

C'est un ensemble de cases qui se suivent, auxquelles on accède par la première, puis la suivante, et ainsi de suite jusqu'à la dernière.

Une cellule de la liste contient l'adresse de la suivante, c'est pour cela que l'on les appelle les listes chaînées.



Un exemple, en prenant pour convention le stockage à l'envers du nombre dans la liste, c'est à dire que le premier sera l'unité puissance zéro, on aura, pour 51987:

La liste correspondante pointera sur le 7, plus facile à utiliser pour additionner deux listes, mais on peut choisir un autre sens. L'avantage des listes étant que, à chaque fois que l'on a besoin d'une nouvelle case, on la crée. Voyons la déclaration d'une liste :

```
Type
    Liste      = ^Cellule;
    Cellule = Record
        Caract    : Char;
        Suivant : Liste;
    End;
```

Il s'agit d'un pointeur sur un type record, contenant le caractère, ou un byte, suivant son goût personnel, et l'affichage. Ensuite, il faut des instructions pour initialiser une liste :

```
Function Init : Liste;
Begin
    Init := Nil
end;
```

Et d'autres pour créer le premier élément. On procédera ainsi:

```
var A : liste;
begin
    a := Init;
end;
```

La liste est initialisée, pour l'allocation, elle se fera toute seule, lors de la première opération, du genre A := CONS ('0', A);, puisque tout est traité avec des caractères par la suite. Voyons cette fonction :

```
Function Cons (Element : Char; L : Liste) : Liste;
Var
    Nouv_Liste : Liste;
Begin
    New(Nouv_Liste);                              (* on cree la cellule
*)
    Nouv_Liste^.Caract := Element; (* on met le caractere
*)
    Nouv_Liste^.Suivant := L;            (* on prend l'adresse envoyee comme *)
    Cons:=Nouv_Liste;                            (* nouvelle case vide potentielle
*)
                                                                  (* et on rattache
le tout                      *)
End;
```

Bien évidemment, on ne peut rajouter une case qu'en tête de liste, d'où le besoin d'établir des conventions, comme stocker le nombre à l'envers, et la première case lue sera celle des unités puissance 0.

Une autre fonction nécessaire est savoir si la liste est vide, si c'est le cas, il n'y a rien à traiter :

```
Function Liste_Est_Vide(L : Liste) : Boolean;
Begin
    Liste_Est_Vide := (L=Nil)
End;
```

Maintenant, pour extraire un nombre, on ne peut extraire que la queue, je m'explique, la tête de la liste pointe sur NIL, au cas où on voudrait rajouter une case, et donc, comme le montre le graphe, c'est la queue de la liste qui contient l'adresse de la cellule suivante et ainsi de suite jusqu'à la tête.

Pour parcourir la liste, on lira la queue, puis on passera à la case suivante... Il n'y a pas de parcours en sens inverse possible, car on ne garde pas trace de la cellule précédente, ce qui est d'ailleurs inutile dans le travail sur les nombres par les listes.

Voici donc comment lire la queue de la liste :

```
Function Premier (L : Liste) : Char;
Begin
    Premier := L^.Caract
End;
```

Et comment la retirer pour lire la case suivante :

```
Function Reste (L : Liste) : Liste;
Begin
    Reste := L^.Suivant
End;
```

Maintenant, pour admirer vos essais, voici comment afficher une liste :

```
Procedure Affiche_Liste(L : Liste);
Begin
    If Not Liste_Est_Vide(L) Then
    Begin
        Write (Premier(L));
        Affiche_Liste(Reste(L))
    End
End;
```

Il arrive qu'au bout d'un moment on veuille savoir le nombre d'éléments d'un liste, voici comment y arriver :

```
Function Longueur (L : Liste) : integer;
Var
    TAILLE : Integer;
Begin
    TAILLE := 0;
    Repeat
        L := Reste(L);
        TAILLE := TAILLE + 1;
    Until Liste_est_vide (L);
    LONGUEUR := TAILLE;
End;
```

Et à cause de la convention de stockage, il arrive parfois que, par la récursivité, que la liste soit renvoyée à l'envers, et donc il faut l'inverser :

```
Function Miroir (L : liste) : Liste;
Var
    L_inv : Liste;
Begin
    L_inv := init;
    While not Liste_est_vide(L) do Begin
        L_inv := cons(premier(L),L_inv);
        L          := reste (L);
    End;
    Miroir := L_inv;
End;
```

Le renversement à été fait de manière linéaire, et ne pouvait pas être fait de manière récursive ici à cause des listes. Voyons maintenant comment comparer deux listes, ci-dessous ce test, on envoie A et B et on veut la comparaison :

```
Function Test_A_Plus_Grand_Que_B (NA,NB: Liste): Integer;
Var
      B: Integer;
      (* NB : Valeurs possibles de B:
                     0: Egalité de NA et NB
                     1: NA>NB
                     2: NA<NB                                    *)
Begin
    If Longueur(NA)>Longueur(NB)
        then B:=1
        else If Longueur(NA)<Longueur(NB)
            then B:=2
            else If Longueur(NA)=Longueur(NB)
                then begin
                    NA:=Miroir(NA);
                    NB:=Miroir(NB);
                    While (premier(NA)=premier(NB))
                        and (not Liste_est_vide(NA)) do Begin
                            NA:=Reste(NA);
                            NB:=Reste(NB);
                        End;
                    If (Liste_est_vide(NA) and Liste_est_vide(NB))
                        then B:=0
                        else If ord(premier(NA)) > ord(premier(NB))
                            then B:=1
                            else If ord(premier(NA)) < ord(premier(NB))
                                then B:=2
                    end;
    Test_A_plus_grand_que_B:=B;
End;
```

Plus compliqué qu'avec les tableaux, et plus lent oui, mais les inconvénients ont des fois des avantages. Ci-après, une fonction qui extrait p éléments de la liste N, ça peut servir des fois :

```
Function XTRACT (N : Liste; P : Integer) : Liste;
Var
    RES: Liste;
Begin
    RES := Init;
    N      := Miroir(N);
    While not(Liste_est_vide(N)) and (P>0) do Begin
        RES := Cons(Premier(N), RES);
        N      := Reste(N);
        P      := P-1;
    End;
    XTRACT:=RES;
End;
```

Après ces quelques amusements, le gros du boulot, l'addition de deux listes, ici c'est très simple et clair, grâce aux listes. En deux mots, la procédure principale reçoit les deux listes NA et NB, teste la nullité, dans le cas contraire additionne NA et NB dans NC, et quand une des deux listes est vide, envoie le traitement à la procédure imbriquée fin_d_add, qui finit le boulot, mais regardez plutôt :

```
Function Addition (NA, NB : liste) : liste;
```

```
Var
    NC            : Liste;
    retenue : Integer;
    a,b           : Integer;

Procedure fin_d_add(var ND:liste);
Begin
    While not (liste_est_vide(ND)) do Begin
        NC            := cons(chr( ((ord(premier(ND))+retenue-zero) mod
10)+zero),NC);
        retenue := (ord(premier(ND))+retenue-zero) div 10;
        ND:=reste(ND);
    End
End;

Begin
    If Liste_est_vide(NA) then NA:=cons('0',NA);
    If Liste_est_vide(NB) then NB:=cons('0',NB);
    NC            :=init;
    retenue :=0;
    While not ((Liste_est_vide(NA)) and (Liste_est_vide(NB))) do
        If ((Liste_est_vide(NA)) and (not(Liste_est_vide(NB))))
            then fin_d_add(NB)
            Else If ((Liste_est_vide(NB)) and (not(Liste_est_vide(NA))))
                then fin_d_add(NA)
                Else Begin
                    A              := ord(premier(NA))-zero;
                    B              := ord(premier(NB))-zero;
                    NC            := cons( chr( ( (a+b+retenue) mod 10)+zero), NC);
                    retenue := ( (a+b+retenue) div 10 );
                    NA             := reste(NA);
                    NB             := reste(NB);
                End;
    If retenue<>0 then NC := cons(chr(retenue+zero),NC);
    Addition:=Miroir(NC);
End;
```

C'est quand même plus simple que les tableaux non ? Du moins c'est plus clair. Continuons dans le concret, voici la différence de deux listes. Pour simplifier la lecture, on en a fait 3 procédures imbriquées, la première principale, cherche la plus grande, car la différence est positive toujours, et traite la nullité. Après, elle appelle SOUSTRACT, qui fait la soustraction de cellule à cellule jusqu'à ce que une des deux listes soit vide, et c'est la procédure FIN_D_SOUS qui finit le boulot. Admirez :

```
Function SOUSTRACTION (NA, NB : liste) : liste;

Function SOUSTRACT (NA, NB : liste) : liste;
Var
    NC            : Liste;
    Retenue : Integer;
    a,b,ret : Integer;

(********** Fin de la soustraction quand liste vide **********)
Procedure FIN_D_SOUS(Var ND: liste);
Var
    C: Integer;
Begin
    While not (liste_est_vide(ND)) do Begin
        RET :=0;
        C      :=ord(premier(ND))-zero;
        if C<retenue then Begin
            C:=C+10;
            Ret:=1;
        End;
        NC            :=cons(chr(c-retenue+zero),NC);
        retenue :=ret;
        ND            :=reste(ND);
    End
End;
(* Fin de FIN_D_SOUS *)

(* debut de SOUSTRACT *)
Begin
    NC            :=Init;
    Retenue :=0;
    A              :=0;
    B              :=0;
    While not ((liste_est_vide(NA)) and (liste_est_vide(NB))) do
        if liste_est_vide(NB)
            then FIN_D_SOUS(NA)
            else Begin
                A              := ord(premier(NA))-zero;
                B              := ord(premier(NB))-zero;
                ret          := 0;
                If A<(B+retenue) then Begin
                    A      := A+10;
                    RET := 1;
                End;
                NC            :=cons(chr(a-(b+retenue)+zero),NC);
                retenue :=ret;
                NA            :=Reste(NA);
                NB            :=Reste(NB);
            End;
    Soustract:=Miroir(NC);
End;
(* Fin de SOUSTRACT *)

Var
    T : Integer;
Begin
      T := Test_A_Plus_Grand_Que_B(NA,NB);
      If T=0
          then Soustraction:=cons('0',nil)
```

```
          Else If T=1
              then Soustraction:=Soustract(NA,NB)
              Else If T=2
                  then Soustraction:=Soustract(NB,NA);
End;
```

Puisque nous voilà lancés, maintenant voyons la multiplication d'une liste par une constante, chose fort peu aisée, et on remarque ici un retournement du résultat de l'opération :

```
Function MultC (L1 : Liste; d : integer) : Liste;
Var
      Multiplic : Liste;
      Retenue      : Integer;
Begin
    If (d<>0) then Begin
        Multiplic:=init;
        Retenue:=0;
        While not (Liste_est_vide(L1)) do Begin
            Multiplic := cons( chr((((ord(premier(L1zero)*d+retenue)
                                    mod 10)+zero), Multiplic);
            Retenue      := ((Ord(premier(L1))-zero)*d+Retenue) div 10;
            L1                := Reste(L1);
        End;
        If (Retenue<>0)
            then Multiplic:=cons( chr (retenue+zero), multiplic);
        MultC:=Miroir(Multiplic);
    End
    Else MultC := cons('0',Nil);
End;
```

Voici ensuite la multiplication généralisée de deux listes, qui utilise la procédure précédente, et renvoie le tout dans une liste. On procède en prenant le dernier de la plus petite liste, que l'on multiplie par l'autre liste, on met ça dans L_partielle, puis ce nombre étant stocké à l'envers par convention, pour le rajouter à L_totale, il suffit de lui rajouter autant de zéros que son rang :

```
Function Multiplication (N1, N2 : Liste) : Liste;

Function Multiplicat (L1, L2 : Liste) : Liste;
Var
    L_totale        : Liste;
    L_partielle : Liste;
    I                       : Integer;
    N                       : Integer;
    Puiss              : Integer;
Begin
    L_Totale        :=Init;
    L_partielle :=Init;
    Puiss              :=1;
    For I := 1 to Longueur (L2) do Begin
        L_partielle := (MultC(L1,( ord(premier(L2))) - zero));
        For N:=1 to (Puiss-1) do Begin
            L_partielle:=Cons('0',L_partielle);
        End;
        Puiss := Puiss+1;
        L_totale := Addition(L_totale,L_partielle);
        L2 := Reste(L2);
    End;
    Multiplicat:=L_Totale;
End;

Begin
    If longueur(N1)<Longueur(N2)
        then Multiplication:=Multiplicat(N2,N1)
        Else Multiplication:=Multiplicat(N1,N2);
End;
```

Ici plus bêtement, on multiplie p fois N par 10 pour le mettre à la puissance p voulue :

```
Function PUISS_10 (N : Liste; NB_ZEROS : Integer) : Liste;
Var
      I : Integer;
Begin
    For I:=NB_Zeros Downto 0 do
        N:=Multiplication (N,Cons('1',Cons('0',nil)));
    Puiss_10:=N;
End;
```

Ca c'était le digestif avant le second plat de résistance : la division. Essayez donc de comprendre tout seul, la mise en page me fait damner...

```
Function DIVISION (NA,NB:liste; Var RESTE: Liste):liste;

Function Divis (N1,N2: Liste; RESULT : Liste; Var RESTE: Liste) : Liste;
Begin
 If TEST_A_PLUS_GRAND_QUE_B (Xtract(N1,LONGUEUR(N2)),N2)=2 then
      If TEST_A_PLUS_GRAND_QUE_B (Xtract(N1,LONGUEUR(N2)+1),N2)=2 then
      Begin
          Divis:=Miroir(Result);
          Reste:=N1;
      End
      Else Divis:=Divis(
                                        Soustraction(N1,
                                        Puiss_10(N2,
                                        Longueur(N1)-Longueur(N2)-1)
                                        ),
                                        N2,
                                        Miroir(Addition(miroir(Result),
                                        Cons('1',nil))), reste)
      Else Divis:=Divis (
                                          Soustraction(N1,
                                          Puiss_10(N2,
                                          Longueur(N1)-Longueur(N2))
                                          ),
                                          N2,
                                          Miroir(Addition(Miroir(Result),
                                          Cons('1',nil))), reste)
End;

Begin
    if TEST_A_PLUS_GRAND_QUE_B(NA,NB) = 1
    then Division := Divis(NA, NB, NIL, reste)
    else If TEST_A_PLUS_GRAND_QUE_B(NA,NB) = 2
        then Division := Divis(NB, NA, NIL, reste)
        else Division := NIL
End;
```

Vous venez d'admirer un grand moment de récurrence de listes, de rappels de fonctions. Ca serait vraiment très long à schématiser pour vous expliquer, mais je garde un grand espoir de vous faire un jour un cours de numération. Et pour finir, le modulo de A par B, du gâteau :

```
Function MODULO(NA, NB : Liste) : Liste;
Var
    DIVI, RESTE : Liste;
Begin
    Divi      := init;
    Reste    := init;
    Divi      := Division(NA,NB,RESTE);
    Modulo := reste;
End;
```

Conclusion

Vous voici tout clef en main, vous n'avez plus qu'à faire du copié-collé, comme moi avec l'article précédent. Dans un futur proche, je vous ferai peut-être une description des méthodes de tri avec listes et tableaux, si j'en

trouve le courage.

Pour aller plus loin ... je signale qu'il existe une fonction pré-existante en C utilisable par LIST.H, et qui dans le même genre definit des arbres binaires... Si cela vous intéresse, je vous en parlerai. Sachez seulement qu'un arbre binaire est une cellule qui pointe sur deux autres cellules et ainsi de suite, ce qui en le représentant sur du papier donne une sorte d'arbre.

Autre chose, si vous faites un travail à base de suites de Fibonnacci, contactez moi, j'ai un rapport de 40 pages au moins et des programmes à ce sujet.

Et si cela vous intéresse, je pourrais vous taper tout le cours de niveau IUT, BTS et 1e année de supérieure, mais plebiscitez moi...

Famous last words

Bien sûr, si certains bouts de programmes ne marchent pas, où si vous avez mieux, je suis prêt à tout accueillir pour améliorer ce document.

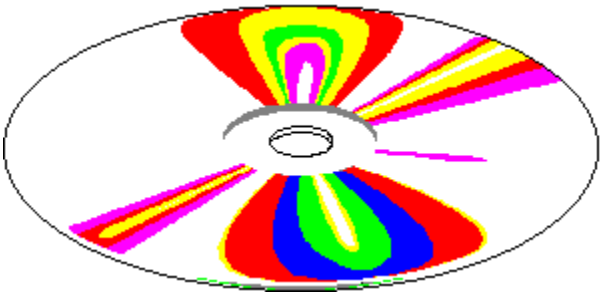
Remerciements

Murdoc, mon binôme à qui on doit ces merveilleuses adaptations de ce que nous avions produit en TP pendant l'année. A bien y relire, j'ai des doutes sur la division... Puisse-t-il réussir mieux son devenir.

A.Lofficier, la diligente bonne fée de notre école, qui aide de son mieux ceux qui sont en difficulté, et qui me force à "produire du sens". et comme envoi, A celle qui ...

L'intégrale des CD-ROMs

Psychess [Juin 95]



PRESENTATION

Les fameuses compilations de 10 CD et plus valent-elles la peine d'être vécues ? Heureux possesseurs de lecteur CD, vous vous êtes déjà laissé tenté par l'achat des packs 5 ft 10, Mégamédia, Explorer, Treasure, Mégapak, CD ROM Bunde,...Ben, nous aussi à la rédaction du REPORTER, on a jeté les pieds dans le plat. Ne pouvant tous les essayer (le trésorier aurait peut-être accepté si on avait fait 50/50 sur la redistribution des 82 CD totalisables...) on a (le "on" c'est moi tout seul : ça fait moins narcissique !) sélectionné le 5 ft 10 pak volume II. Espérons qu'il soit représentatif ou si non le pire de tous...

Le prix:

199 Frs (chez "mon" revendeur au 10/06/95) [231 Frs avec les frais de port et emballage.]

Le contenant:

Pochette plastique banale de 11 compartiments à CD à face visible. Plié, c'est de la taille d'un boitier 4 CD (comme le CICA Windows). C'est donc un bon rapport <<encombrement/protectionCD>>. Tant mieux pour les bordéliques invétérés. Ceux qui désirent exposer fièrement leur trouvaille pourront clouer cette pochette au mur : une encoche à l'entête est déjà perforée. Elle sera plus utile aux revendeurs.

Le contenu:

10 CD de 12 cm de diam. et 1 mm d'épai.: vérifiez que ça rentre ! 2 dépliants de 50 sur 75 cm style carte de France. Pas pratique lorsque, après 30 longues heures de consultation des CD et 2 ou 3 nuits blanches, vous devez ne pas laissez de trace de passage sur l'ordinateur de Papa. Faut dépliez ces grandes feuilles car elles sont censées remplacer le "Manuel de l'utilisateur". En fait sur l'une, il y a 50% de pub pour avaler X autres CD. Bref vous avez le pressentiment que vous passerez pas 3 heures à lire la documentation (qui, bien sûr, est exclusivement en anglais US). La présentation des produits est suffisamment claire, répartie en 4 principaux thèmes (installation, menus de l'interface, informations générales et astuces). On a déjà vu bien pire dans le style économique et là on a même le droit à la couleur: différentes nuances de violet et de noir. Des rubriques "trucs et astuces" appréciables sont mises en relief.

MULTIMEDIA JUMPSTART (Microsoft)

Sous Windows

Oui, Oui ! A ce prix là on peut encore avoir le label Microsoft. Si il est vrai que ce CD est conçu à l'origine pour les programmeurs (avec une préférence pour le C et le multimédia) il intéressera aussi le débutant avec une grosse quantité de vidéos (avi,flc) son (wav) et images (bmp).

Pour faire l'exploration de ce CD on dispose du "Browser and Jumpstart controller": une télécommande à 10 boutons et un écran d'édition. C'est bien mais insuffisant car (comme le conseillent certains textes de présentation des démos) il faut repasser par le gestionnaire de fichiers pour lancer la plupart des éxécutables. Vu l'impressionnante pleïade des programmeurs du produit il y a de quoi s'étonner. Quant aux découvertes elles sont très variées et vous proposent tour à tour de créer,animer,écouter, tester, installer, documenter, scanner, imprimer,...Bref le multimédia est bien couvert à toutes les sauces.

Le bon côté des choses:

Test de lecteur CD, un multiconvertisseur (audio/midi/image), un dossier technique introductif aux polices True Type, un didacticiel musical style dessin animé (pour les enfants...).

C'est dommage:

Les démo les plus réussies sont bridées comme le Lenny's Music Toons. C'est pas étonnant: à ce prix là, Microsoft peut se permettre de nous refiler un CD bourré de pub !

Verdict:

vraiment bon même pour les non-programmeurs. N'étant pas qualifié pour juger de la qualité des sources modèles, mon appréciation reste limitée aux démos.

Des chiffres:

Ext	Nbre	Mo
ASM	29	0.524
AVI	105	247
BAS	14	0.470
BMP	143	13.8
C	115	1.7
DLL	119	10.3
EXE	85	14
FLC	24	32.5
VBX	26	1.3
WAV	91	30.4

Ext.= Extension du fichier (ASM: asssembleur, AVI: vidéo *.avi)

Nbre = Nombre total de fichiers dans CD

Mo = Mégaoctets utilisés au total

Richesse : 

Esthétique :

Documentation :

Facilité d'installation :

Facilité d'utilisation :

Rapport qualité prix :

BATTLE CHESS (Interplay)

Sous Dos & Windows

Destiné aux amateurs du jeu d'Echec, il présente sous une forme attractive (combats animés en VGA 256 coul.) les subtilités du jeu qui a passionné des générations de rois, de philosophes,...

Au XVIè siècle, surtout en Italie, au cours de certaines fêtes, se donnaient en spectacle des parties d'Echec dont les pièces étaient représentées par des personnages vivants. Cela se fait encore de nos jours (si, si en tant que passionné, j'en témoigne !) et Battle Chess (passé à la postérité depuis longtemps) en a fait une version informatique très connue.

Ce qu'on apprécie:

◆ Les fonctions telles que:
charger, sauvegarder la partie, obliger l'ordinateur à jouer (dès les niveaux 5/6 votre patience sera mise à rude épreuve... vos capacités intellectuelles aussi!), reprise et suggestion de coups (attention: c'est de la triche!), possibilité de jouer par modem (un beau gadget), encore par simple cable sur "ports série" pour pouvoir jouer discrètement avec les collègues du bureau.
◆ Le son est de bonne qualité avec des bruitages pour les déplacements des personnages, leur combat. Il y a aussi une musique de fond dans un style médiéval et le tout correctement synchronisé.
◆ Les combats diffèrent selon les pièces qu'ils opposent. Leur fluidité est bonne (je sais: c'est tout de même un DX4/100 qui fait le test...). Et si à la fin, tous ces divertissements vous ennuient, Basardez la 3D, coupez le son et entreprennez de profondes réflexions dans un calme religieux.
◆ Tout est fait pour casser l'image vieillie ou les préjugés poussiéreux attribués aux Echecs : le sérieux est renversé par les attitudes et prises de paroles humoristiques des personnages. Il ne faudra pas s'étonner de voir le roi (apparemment vieillard sénile sans défense) descendre son adversaire avec une arme à feu dernier cri.
◆ Si certains ne savent pas y jouer (ils devraient s'y mettre, avec Battle Chess, sans trop d'effort), un didacticiel est prévu. Chaque personnage y présentera son histoire, son déplacement et ses principales qualités. Toujours avec quelques pointes d'humour.
◆ Sur le CD, vous avez le droit à 2 versions: DOS (1) et WINDOWS (1.1) Celle sous Windows est plus riche dans les possibilités de paramétrage des animations. Et sur la machine de test, elle s'est révélée plus rapide pour les images mais aussi pour la vitesse de calcul des réponses. Le son y était même sensiblement meilleur. C'est assez
surprenant !

On regrette quand même que:

◆ Sous DOS et sur un 386 ça risque de paraître un peu lourd
◆ L'apprentissage du jeu soit un peu trop superficiel. Il faut que vous acceptiez de vous faire ridiculiser de nombreuses fois pour comprendre comment débuter une partie puis gagner. De cette façon, on est obligé d'observer l'ordinateur jouer. Ceux qui ne connaissent strictement rien aux Echecs risquent bien de fuir le jeu avant même d'y prendre goût.
◆ La notice d'utilisation ne comble pas ce manque. Ceux qui ne veulent que l'essentiel seront à l'honneur. Tant pis pour les autres.
◆ La ceinture de chasteté n'est pas fournie.
◆ Cette ambiance moyen-âgeuse et film de chevalerie me fait écrire n'importe quoi !

Résultat des courses:

Bon produit qui n'a pour objectif que de divertir. Dire que ça coûte 20frs par CD...En cadeau, vous trouverez 12 sharewares et 1 démo des autres jeux d' Interplay.

Richesse :

Esthétique :

Documentation :

Facilité d'installation :

Facilité d'utilisation :

Rapport qualité prix :

SPACE QUEST IV (Sierra)

Sous Dos

C'est bien ce jeu là qui a permis à un jeune programmeur de talent de devenir le riche patron de Sierra. Depuis la version I du célèbre jeu d'aventures réalisée en 1986, il y a eu plusieurs évolutions marquées par celles de l'informatique en général. C'est avec le N°IV qu'appraissaient les voix digitalisées.

Roger WILCO, le héros (ça dépend avec qui...) n'a pas revu sa planète Xénon depuis la version N°II. Sur la planète Magnéthéus, alors que vous bavardez tranquillement dans le bar du coin, deux mutants vous prient de bien vouloir les suivre. Pas de problème: ils sont armés jusqu'aux dents, ils vous surplombent du haut de leur 2m, et vous aviez fini votre verre. Ils sont là pour vous faire la peau. Ca commence mal! Deux inconnus surgissent et vous sortent des griffes des 2 mal intentionnés. Sur ce, vous êtes téléporté vers une cité détruite, dont il ne reste plus que des ruines et des zombies... Les décors et la musique sont lugubres à souhaits. Et c'est dans ces conditions qu'il faut chercher à ne pas sortir de cette histoire les pieds devants.

Ce qui est plaisant:

◆ La possibilité de faire transcrire à l'écran toutes les remarques qui sont faites à l'origine par voie audio. Ceux qui écoutaient leur prof d'anglais adorée comprendront que l'accent américain n'est pas très "British". La lecture pallie aisément le petit problème.
◆ Sur le dépliant vous avez le droit à des "Cheat note": on vous dit tout pour éviter les premières embûches pour commencer l'exploration dans de bonnes conditions. Franchement, c'est pas de trop lorsque vous êtes pressé.
◆ L'humour: quand vous débarquez dans cette cité sordide, c'est le moindre des réconforts. Si vous avez l'imprudence de tester les commandes sur votre personnage (évidemment, je m'y suis collé !), vous allez au devant de remarques narquoises. L'outil "nez" donna: "Aaaaaah! The aroma of several adventure games emanates from your person".

* des bonus:

◆ On a le droit à une sélection de 4 démos de jeux d'aventures.
◆ Un jeu spécial modem

Ils auraient pu faire un effort sur:

◆ La documentation sur CD est vraiment ridicule: pas un mot sur l'histoire, le but du jeu n'est même pas évoqué. Si c'est la première fois que vous débarquez dans la série Space Quest, à vous de vous débrouillez (!). Heureusement, le dépliant vous aide pour les premiers sentiers.
◆ La musique est vite redondante.
◆ Pour les graphismes, c'est du "Ouais, bof...".
◆ Les animations sont simples.
◆ Les traductions: il n'y en a pas! On regrette de n'avoir pas été plus sérieux durant les cours d'anglais. On va être obligé de s'y remettre.

Finalement:

Ce n'est pas mon style de jeu préféré donc ma position est particulièrement subjective. Il faut admettre que si il y a encore 4 ans il fut, chez les adeptes, admiré, en 1995, il ne mérite plus le même engouement. Preuve en est qu'il se retrouve dans ce pack bon marché.

Richesse : 

Esthétique :

Documentation :

Facilité d'installation :

Facilité d'utilisation :

Rapport qualité prix :

SHERLOCK HOLMES I, II & III (Icom Simulation)

Sous Dos

Voilà encore un jeu qui fait appel à votre perspicacité et votre concentration. Dans la peau de l'ingénieux détective HOLMES, votre tâche est de récolter le maximum d'informations et d'indices dans un minimum d'investigation. A votre disposition: des dossiers, un carnet d'adresses, un bloc-note, le journal "The Times", un informateur, un carrosse pour vos déplacements et l'incontournable Dr. WATSON.
"Miss a single query and you will be sent back for more clues" Vous êtes prévenu: ne laissez rien passer...Malheureusement on croule sous une avalanche d'informations de toutes sortes: 16 pages du "Times" à consulter en témoignent. Sans compter les innombrables dialogues avec WATSON, la bonne centaine d'adresses à vérifier, les dossiers à compulser: c'est alors que vous comprennez que ce métier n'est pas fait pour vous. Si vous résolvez le premier épisode, "la malédiction de la momie", où quelques archéologues sont morts mystérieusement après la découverte d'une momie vieille de 4000 ans (rien que ça!), vous pourrez affronter le cas de "l'intriguante meurtrière" (une sombre affaire de moeurs pas très catholiques)ou encore le cas du "Soldat de plomb".

De bonnes pistes:

◆ Le type de jeu même est intéressant: la recherche d'indices et la mise en accusation des suspects vous oblige à vous investir pleinement. Lorsque vous trouvez la solution vous avez de quoi flatter votre orgueil pour quelques jours.
◆ Le personnage SHERLOCK HOLMES est rusé et très populaire. On a du plaisir à le revoir dans un jeu informatique, alors que de nombreux livres, films et dessins animés lui furent consacrés.
◆ Les cas sont introduits par des vidéos.
◆ La documentation: enfin un fichier texte qui recèle le minimum requis pour pouvoir démarrer le jeu confortablement.
◆ La simplicité d'utilisation est renforcée par un didacticiel de présentation des outils de recherche et des principales personnes (et institutions) à rencontrer.

Restent des obstacles:

◆ Le principal est évidemment la langue. Non pas que leur anglais soit bâclé (bigre, c'est bien le contraire!), mais les dialogues sont exclusivement parlés (j'avoue: c'est normal). On doit faire entièrement confiance à nos capacités auditives lors des échanges avec le bon vieux WATSON. C'est pas de la tarte croyez-moi !
◆ Des bruits de fond (cheval, carrosse, bavardages,...) agrémentent les dialogues. C'est bon pour le réalisme d'un dialogue de rue et mauvais pour la compréhension linguistique.
◆ La plupart des "vidéos" sont trop courtes et se réduisent à une image figée en N&B avec des échanges verbaux. Les introductions promettaient plus.
◆ La musique est quasi absente durant l'enquête alors que l'intro du jeu donnait l'ambiance "suspense sur un quai un soir de brume..." C'est bon pour le moral !
◆ Les décors sont grossièrement réalisés par rapport aux banales images de synthèses et 3D mappées de notre univers multimédia actuel.

Alors, finalement ?

Le sujet est captivant mais l'anglais n'est peut-être pas votre... tasse de thé (j'ai osé). On a tout de même 3 épisodes complets sur ce CD à bas prix, c'est pas négligeable. Même si vous ne jouez pas avec, vous pourrez toujours l'utilisez pour parfaire votre prononciation avant de paraître ridicule lors de votre prochain séjour linguistique (à condition d'aller à Londres et non pas à New-York).

Richesse : 

Esthétique :

Documentation :

Facilité d'installation :

Facilité d'utilisation :

Rapport qualité prix :

HOME MEDICAL ADVISOR V.3.0 (Pixel Perfect)

Sous Windows

Ce programme vous donne de multiples informations à propos de vos malaises, blessures, symptômes, drogues, poisons, régimes, aptitudes. On peut même se faire passer des tests médicaux simples pour savoir si vous en avez encore pour longtemps. Ca c'est la partie "base de données". Du côté "interface" c'est un hypertexte indépendant de son grand frêre intégré à Windows 3.1.

Le but n'est pas de se substituer totalement à votre bon vieux bourreau attitré: le médecin de famille. Par contre, il pourrait soulager le trou de la Sécu en vous évitant d'aller consulter le Doc inutilement. En effet, si votre QI n'est pas particulièrement bas, vous devriez pouvoir diagnostiquer et décider si les 2 heures passsées dans la salle d'attente avec cette grand mêre plaintive qui vous hurle dans les oreilles (elle est dure d'oreille...) sont vitales.

C'est mieux que chez mon toubib:

◆ L'installation offre une toute petite originalité: au lieu de la très classique barre de pourcentages qui se rempli au fur et à mesure on a 11 coeurs qui rougissent les uns après les autres.
◆ Rapidité d'accès aux informations
◆ L'aide intégrée est relativement suffisante.
◆ Les diagnostics se font à travers des questions (comme le ferait le médecin) qui selon les réponses (QCM, Oui/Non) vous redirigent vers d'autres questions puis enfin le verdict. Ces 1400 questions sont accompagnées de 600 illustrations et aboutissent à 400 diagnostics différents.
◆ Exportation de tous les textes sur imprimante ou fichier ascii que vous pourrez importer et reformater dans un traitement de texte.
◆ Même si c'est un dénommé Dr. SCHUELER qui signe ce programme on peut noter qu'il y a eu intervention d'une belle brochette de spécialistes pour apporter certaines garanties (dont se défend officiellement le texte d'avertissement, pas étonnant quand on sait qu'aux USA les procès sont monnaie trop courante...) dans les domaines pointus.

Des contre-indications:

◆ Superficialité du champ d'investigation possible.
◆ Besoin d'un bon gros dictionnaire de qualité pour la traduction et compréhension d'un texte bourré d'indications techniques qui ne font pas partie du bagage de base de Mr. TOULEMONDE.
◆ Problèmes de conversion des unités pour les mesures intéressantes.
◆ Les illustrations (vectorielles, 16 couleurs) ne permettent pas d'exploitation médicale: rares sont les dessins anatomiques précis.
◆ Le tout tient sur 5 Mo peut être pour pouvoir tout mettre sur le disque dur mais les 600 Mo restant sur le CD auraient pu être exploités pour des images de qualité ou même vidéo.
◆ Suite logique de la remarque précédente, on se trouve dans un hypertexte proprement présenté mais rudimentaire. Un programme hyperMEDIA aurait été nettement plus agréable pour ceux qui ont investi plusieurs milliers de Frs dans les périphériques multimédia.

Quel diagnostic final ?

A moins d'être obnubilé par sa santé physique, on n'attribuera à ce logiciel que de rares occasions de consultation. Le contraire pourrait être mauvais signe ! Enfin, les étudiants en médecine s'exerceront à poser les bonnes questions et à donner les bons diagn./conseils. L'obstacle de la langue est, encore, ici, majeur.

Richesse : 


Esthétique :



Documentation :



Facilité d'installation :




Facilité d'utilisation :




Rapport qualité prix :


MOVIE SELECT (Paramount Interactive)

Sous Windows

Les fanas du grand écran devraient aimer cette base de données de 44000 films qui renseigne sur les acteurs, le metteur en scène, le thème, les prix reçus, la durée , la date, etc...C'est une très belle interface en 640x480 et 256 coul avec liens hypertextes. L'une des principales attractions de ce programme, pompeusement appellée "Expert System Technology", va vous recommander de voir des films sélectionnés d'après vos préférences sur un échantillon donné.

A voir absolument:

◆ L'interface est colorée, claire et précise. (Quicktime)
◆ Recherche multiple par titre, acteur, metteur en scène, listes alphabétiques, genre et par style. Sans oublier les actrices bien-sûr!
◆ Possibilité d'impression
◆ "fuzzy logic, a type of artificial intellligence" : en "2" minutes vous obtenez une présélection personnalisée de films qui devraient vous plaire.
◆ 14 vidéos dont 12 bandes annonces de films sortis aux USA en 1992 ce qui représente 181 Mo sur le total de 206 Mo sur CD
◆ On trouve assez facilement des films français même certains n'ayant qu'une portée hexagonale; et ce, sûrement grâce aux acteurs ayant acquis (après avoir servi des navets) une renommée internationale.

Ca manque de films et d'infos !

◆ Non, ce n'est pas la richesse des 44000 films qui est remise en cause (il faut s'accrocher pour en faire entièrement le tour!) mais ce sont les informations sur chaque film: 1 à 2 phrases pour le sujet, pas de critique, pas d'affiche reproduite, pas de musique, pas de photos... Ca fait pauvre !

◆ Quelques musiques de films pendant la consultation des données auraient été appréciée pour mettre de l'ambiance: une base de données sur le cinéma ne devrait pas se réduire, pour un non spécialiste, à de ternes lectures qui plus est...sommaires. Il faut donner envie de décoller de sa chaise au petit informaticien obsédé par ses petits écrans (pour preuve le témoignage de KIKI la Rorette, REP 3).

◆ Le module d'intelligence artificielle (plus artificiel qu' intelligent d'ailleurs...) donne des résultats aléatoires pas toujours convaincants. Y aurait-il une part abondante de subjectivité non maîtrisée ?

◆ Pas de possibilité d'actualiser la base: il ne faudra plus chercher les films au ciné mais à la vidéothèque du coin.

La scène finale:

Si c'était un film il serait sans parole, sans musique, sous-titré, avec de nombreux acteurs mais dont l'interprétation ne serait qu'au niveau de banals figurants: un film spirituel superficiel. Bref, ça peut pas être N°1 au box office américain. J'exagère la critique: comme beaucoup de long-métrages, il ne pêche que par l'excès d' exigence du spectateur...

Richesse :

Esthétique :




Documentation :



Facilité d'installation :




Facilité d'utilisation :




Rapport qualité prix :



2000 FONTS (Fantazia)

Sous Dos & Windows

2000 polices au format True Type, Type 1, réparties dans 37 répertoires principaux chacun subdivisés en 3 sous-répertoires nommés \pictures, \truetype et \type1. Nous avons à notre disposition 6 formats de fontes: AFM, CFG, INF, PFB, PFM et le standard Windows TTF (le mieux achalandé). Pour cette collection, on nous a programmé un viewer DOS (2 syllabes au lieu des 6 du mot visualisateur: vive l'économie américaine [sauf si on est obligé de commenter chaque anglicisme....]!) dans le style Turbo pascal 6. Il permet de visionner des images des polices, de les lister et copier. Dans la documentation, on nous donne des conseils pour bien gérer le problème des polices sous Windows. Je vous les donne pour information:

1- Ne pas installer plus de 300 polices (rien que ça ?!)
2- Créer un répertoire réservé aux polices
3- Utiliser le gestionnaire de police pour organiser...les polices.
4- Epargner de l'espace disque en employant les polices à partir du CD.

Quantité et qualité sont ils compatibles ?

◆ Oui ! Ce ne sont pas des fontes shareware bridées: 90 % possèdent les caractères étendus dont les lettres accentuées françaises.
◆ Vous trouverez tous les styles: des plus fantaisistes aux plus classiques.

Mais quoi ?

◆ Il manque un vrai visionneur sous Windows. C'est l'occasion de tester un bon logiciel en libre essai (FontMonster est très riche en possibilités et rapide...Sans vouloir faire de pub à un auteur que je ne connais pas)

Globalement

Quand on recherche une police on est très content d'avoir ce CD à portée de mains. Il est bien fourni dans sa variété de polices et c'est finalement tout ce qu'on lui demande. Donc produit très correct.

Richesse	
Esthétique	   
Documentation	

Facilité d'installation :

Facilité d'utilisation :

Rapport qualité prix :

ARTS AND LETTERS WAR BIRDS (Computer Support Corp.)

Sous Windows

Le CD se compose de 2 principales attractions: le logiciel de DAO ou création graphique (dessin vectoriel) et une base de donnnées sur les avions de guerre.

1. Petit logiciel de Publication Assistée par Ordinateur
d'entrée de gamme. Avec une richesse optionnelle standard correcte, il vous permet de réaliser relativement rapidement (286 s'abstenir !) les cartes d' invitation, de visite, ou autres mises en page un peu plus complexes. Il possède une banque de clip-arts pour plusieurs domaines.

2. Pour la base de données, c'est un programme de type
hypermédia qui nous permet de naviguer entre les sons, images et textes sur la période de 1914 à 1945 de l'aviation de guerre. Il n'a rien d'exceptionnel; son usage sera limité aux inconditionnels des fous volant pour la présentation d'aviateurs, d'engins et leurs bruits... Tout un programme.. Finalement, un peu léger quand même!

Certains atouts

◆ On bénéficie d'une aide en ligne (ou contextuelle) bien faite avec l'utilisation d'icônes et autres images.
◆ Un didacticiel de démonstration se chargera de vous entraîner
◆ Des modèles seront applicables et personnalisables au document.
◆ Fonctions et outils graphiques suffisants pour les dessins vectoriels.
◆ Effets spéciaux pour les textes (arcs, cercles, ombres, couleurs)
◆ Typographie interne utilisable avec les polices déjà installées sur votre ordinateur.
◆ L'import (pic, doc, tif, txt et wmf) et l'export (cgm, csp, eps, tif, scd, wmf et wpg) de fichiers sont satisfaisants.
◆ Exploitation des clip-arts par Drag'n Drop (facile!)
◆ Bonus sur le CD: démos de Spaceage et de "A&L" Express (meilleur).

Des reproches, derrière la façade:

◆ Fonctions de traitement de texte très faibles : pas de multicolonnage, pas de mise en forme élaborée, pas de césure, etc...
◆ Pas de palette flottante
◆ Rigidité de l'interface: personnalisation et paramétrage manquent.
◆ Format des clip-arts propre à ce logiciel donc pas de récupération.
◆ Et, bien-sûr, c'est une version anglophone.

Dans l'ensemble

Si vous possédez un utilitaire de PAO léger (exple:Publisher) ou spécialisé (PageMaker) ou même un graphiste DAO vectoriel (Vincent 2) que vous maitriser vous ne passserez pas beaucoup de temps avec "A&L". Il vous suffira pour faire de rapides menus, cartes de voeux, affiches et bannières.

Des chiffres

Ext	Nbre	Mo			
AVI	4	9	BMP	706	78
TIF	48	32	WAV	704	247

Richesse : 

Esthétique :

Documentation :

Facilité d'installation :

Facilité d'utilisation :

Rapport qualité prix :

PC KARAOKE / FAMILY FUN (Sirius)

Sous Windows et/ou Platine Laser

Ca ne se présente plus: il existe des bars qui ne jurent plus que par ça des chaînes de télé qui en font des émissions, et les éditeurs de logiciels se frottent les mains. Une carte son 16 bits, des hauts- parleurs de 2x25 watts et un lecteur de CD suffiront pour assurer un minimum de confort d'écoute. Branchez votre micro, lancez une chanson et découvrez-vous chanteur...N'infligez pas cet affreux spectacle à vos proches.

C'est du "vrai" karaoké:

◆ Onze chansons: du traditionnel "made in USA" bien de chez eux. C'est dans le style country avec des choeurs pour le refrain. Le dernier tube du dernier tapage médiatique de la Jackson-mania est absent.
◆ Quelques paramétrages: couleur des textes télé-promptés ("computer assisted annotation"), 8 textures de fond, mixage du volume, micro et son "fx".
◆ Si votre Pentium 150 ne le digère pas (un clône nippon ?) vous pourrez toujours l'écouter sur la chaine hifi du salon.
◆ Si vous désirez monter un karaokéthèque, Sirius en propose plein d'autres avec des chansons un peu moins poussiérieuses et ringardes telles que celles des Beatles, Presley, Elton John, Jackson et même Madonna (Ne vous prenez pas pour l'ex-maire de Paris: la culotte ne sera pas fournie à la fin du concert!).

C'est pas un tube du top!

◆ On ne peut pas faire d'arrêt dans le courant de la chanson, balayer exclusivement un passage pour s'entrainer sur une difficulté particulière est impossible.
◆ Pas de version modèle chantée: si vous étiez réellement nul il y a des chances pour que vous le...restiez!

Richesse : 

Esthétique :

Documentation :

Facilité d'installation :


Facilité d'utilisation :




Rapport qualité prix :



ROCK RAP'N ROLL (Paramount Interactive)

Sous Windows

Vous êtes incapables de lire une partition, les cordes de la guitare claquent toutes entre vos doigts, ces derniers se coincent trop souvent entre les touches du piano, les voisins se lamentent dès que le violon couine dans vos bras et pourtant vous vous sentez un réel talent de compositeur-interprète: y a peut-être une solution... Une boite à musiques pré-composées dans une interface colorée et intuitive devrait vous ouvrir les joies de la musique en quelques simples clics de souris. Le Rock, le Rap et le Reggae sont autant de studios diférents, chacun possédant 10 courtes mélodies constituant le corps principal de votre futur morceau. A cela vous rajouterez une sélection de sons parmi 20 échantillons d'instruments et 8 expressions vocales typiques du style choisi.

Y a de quoi devenir musicien:

◆ Simplicité d'utilisation à la hauteur de n'importe quel niveau de connaissance musicale.
◆ Clics et Drag'n Drop généralisés
◆ Interfaces agréables et soignées en 256 couleurs
◆ Utilisation de sons possible par le clavier
◆ Enregistrement de la composition (hypercompact = gain de place)
◆ Enregistrement de sons externes aux studios réutilisables ensuite.
◆ Un exemple musical par style est proposé. En quelques minutes, on peut allègrement dépasser en qualité ces pauvres exemples...

Ne rêvez pas du top 50...

◆ Quand on a testé l'ensemble des 124 sons, on trouve ça très incomplet et limité: la richesse des paramétrage a été sacrifiée sur l'autel de la simplicité. Vous ne pouvez pas tout avoir...
◆ Le CD est vide ! 5.5 Mo seulement, la plus basse quantité de données sur les 9 autres CD...Et pourtant, il exige d'installer et donc d'occuper 10 Mo sur le disque dur, encombrement le plus fort parmi les 9 autres. Tristes records !
◆ Si on est très critique, on signalera une aide un peu trop légère mais de toute façon elle n'est pas, ici, indispensable.

Richesse	
Esthétique	
Documentation	
Facilité d'installation	
Facilité d'utilisation	

Rapport qualité prix :

CONCLUSION

Du côté des AVANTAGES on retiendra évidemment le prix qui est le principal argument de vente. Et puis une certaine abondance et variété des fichiers/programmes proposés couvrant plusieurs domaines d'intéret général. La facilité des installations est plutôt bonne et ne nécessitent pas le recourt à des astuces et contortions d'initiés.
Par contre le gros INCONVENIENT est indiscutablement la langue. Pas une seule phrase en français n'a pu être relevée court de l'investigation. Faut pas être allergique à l'anglais et un bon dico n'est pas de trop. On ne s'étonnera pas non plus de disposer de logiciels de versions anciennes qui par conséquent n'exploiteront que trop rarement les qualités de votre carte son AWE 32, du lecteur CD quadruple vitesses, des enceintes 180 Watts dans votre station Pentium 133 Mhz tout fraîchement achetée à prix d'or. La plupart des CD proposent l'achat d'un produit complémentaire ou de la version la plus récente. Ce qui laisserait dire aux mauvaises langues que ces packs ne sont que des tremplins vers d'autres achats à fortes plus-values à travers la vente d'un stock d'invendables...Pour 200 ou 250 Frs les risques d'être totalement dégouté de cet achat sont très faibles. A 20 ou 25 Frs le CD, même les commandes de disquettes de logiciels de démos bridées deviennent financièrement plus coûteuses. Bref, si ,dans la présentation d'un des 8 ou 10 packs actuellement (au 06/95) existant sur le marché, 1 ou 2 logiciels au moins, semblent satisfaire un besoin, inutile de résister !

Stars

Rixed [Jul/Aou 95]

BARTI ET KARL REALISENT LA PREMIERE BONNE DEMO PC

A l'heure où j'écris ces lignes, l'Assembly 95 viens de se terminer, et toutes les démos sont déjà présentes sur plusieurs BBS dont les sysops se sont fait fort durant la dernière nuit du concours d'écumer le net pour être les premiers à les proposer en download.

Dès la nuit du dimanche à lundi, tout le monde est averti que l'asm95 à été d'un haut niveau. Plus particulièrement, une démo se démarquerait des autres. Une démo à laquelle cet article est presque entièrement consacré. Une démo Française, du groupe Nooon. Une démo intitulée "STARS : Wonders of the World".

Avant même que je vois la démo, une première personne me confia son découragement devant tant de perfection. Le lendemain, une seconde personne m'appela pour la même raison. Incroyable mais absolument vrai.

Il faut dire que depuis deux années, depuis que les Grands s'étaient plus ou moins retirés de la scène (je pense à Future Crew, à Triton, et dans une moindre mesure à Cascada) en laissant derrière eux quelques démos mythiques qu'on croyait inégalables et qui servaient de références universelles, la démoscène PC manquait terriblement de style et d'audace.

Mais en deux ans, les machines devenant de plus en plus puissantes (le dx2-66 VLB avec ET4000w32 est la machine de référence généralement adoptée, et on va gentiment vers le pentium), l'orientation des démos sur PC fut très net : les démos ne mettaient plus en valeur la technicité du code mais juste la beauté visuelle de l'image.

C'est ainsi que les animations apparurent (Cf. Project Angel), à la stupéfaction des autres scènes (Amiga et Atari), et surtout fleurirent des effets de textures 3d à tout vent, avec mapping, "phong", reflets, ombres, et

compagnie, tout ces effets superposés les uns sur les autres, pour rendre tant bien que mal le premier objet 3DS de 1500 faces qui tombaient sous la main des codeurs. C'est le cas de Dope (Complex), qui marqua le début de ce que j'appellerais, pour des raisons évidentes qui n'échapperont qu'à ceux qui n'ont pas vu la moindre démo datant de ces deux dernières années, l'époque "masque" de la scène PC (Epsylon, No!, LifeForm, etc, ont toutes repris ce fameux masque de 710 faces et 410 points
- eh oui, moi aussi j'ai joué avec :)).

Durant ces deux années, de très nombreux groupes ont émergés, et tous ont repris leur objets 3DS, les ont rendu en phong, les ont fait morpher, etc..., dans l'extase générale, dans l'illusion de savoir coder. Alors qu'à l'échelle ST, la plupart des codeurs PC ne seraient que des apprentis lamers. N'ayant jamais eu de ST, je m'imagine être d'assez bonne foi.

Pas étonnant, donc, que STARS fut ressenti comme un anéantissement pour certains, ou comme une démo qui marque l'avènement d'une ère nouvelle par ceux qui recherchaient la Rédemption.

Pour résumer la démo, c'est assez simple, car il n'y à basiquement rien de neuf. C'est juste beaucoup plus beau, beaucoup plus impressionnant, beaucoup plus imposant, et beaucoup plus rapide. C'est juste du bon code. La musique et les gfx sont aussi irréprochables, mais c'est surtout le code qui marque.

Avant STARS, on pouvait faire une routine de mapping en une journée, mapper un objet 3ds en phong, se faire une séquence en deux jours, et se convaincre soi même de savoir coder. Je sais de quoi je parle, je l'ai fait la semaine dernière. Maintenant, ce n'est plus possible. L'objet devra posseder plus de 20000 faces et tourner toujours en plus de 50Hz le "turbo" enlevé. Les faces des objets devront être indénombrables et l'animation ne devra cependant souffrir d'aucun ralentissement.

Bref, beaucoup de "codeurs" qui se sentaient prêt pour sortir une démo vont devoir retourner à l'école.

Pour le reste, voici toujours les résultats officiels de l'assembly...

Rixed, aprenti lamer avec les autres

PC DEMO

1	10 (3646 points)	Noon "Stars : Wonders of the world"
2	7 (1482 points)	Juice "Psychic link"
3	12 (1204 points)	RealTech "DX Project"
4	1 (1039 points)	Capacala "Zweilight Zone"
5	2 (916 points)	Epical "Rebel Mind"
6	3 (896 points)	Miracle "Higher Desire"
7	15 (799 points)	Orange "Television"
8	5 (587 points)	Dubius "Optimal Torque"
9	8 (541 points)	Kosmic "Little Green Men"
10	13 (372 points)	Japotek "Figthing for something"
11	4 (224 points)	Masque "Mystery"
12	6 (175 points)	tArzAn tuotanto "Syllabization"
13	11 (153 points)	Black Rain "Overflow"
14	9 (143 points)	Deus ex Machina "Fever"
15	14 (33 points)	Simplicity "Elegant"

AMIGA DEMO

1	11 (2917 points)	Parallax "ZIF"
2	9 (1950 points)	Pygmy Projects "Logic"
3	12 (1366 points)	Stellar "Miracles"
4	10 (1224 points)	Silents "Fruit Kitchen"
5	2 (980 points)	Juliet and Case "C42"
6	7 (902 points)	Complex "Dive"
7	8 (457 points)	Fanatic "Hate 2"
8	6 (356 points)	Balance "Embryo"
9	4 (317 points)	Zymosis "Wc goes to wc"
10	3 (281 points)	Embassy "Thrilled"
11	1 (235 points)	Abyss "High Anxiety"
12	13 (217 points)	Black Lotus " "
13	5 (115 points)	Domination "Domination's Dentro"
14	15 (27 points)	""
15	14 (26 points)	""

PC 64K INTRO

1	4 (2750 points)	Wild Light "Drift"
2	10 (1899 points)	Coma "Stickman's World"
3	15 (1364 points)	Complex "Bill G Force"
4	14 (1219 points)	Jamm "Nation Zero"
5	2 (1112 points)	Halcyon "Detour"
6	13 (1041 points)	Valhalla "Believe"
7	1 (527 points)	Mist "Alchymid"
8	6 (506 points)	Halo "Reality Impact"
9	3 (471 points)	Symptom "Camera"
10	11 (364 points)	Abaddon "Pied"
11	9 (358 points)	Epsilon "Dream"
12	7 (271 points)	Hazard "Cocaholic"
13	12 (225 points)	Anarchy "Tam"
14	8 (194 points)	Woodpeckers from Mars "No Class"
15	5 (185 points)	@ "Byte me"

AMIGA INTRO

1	2 (2312 points)	Sonik "FAD"
2	3 (1933 points)	Hirmu "Hauki"
3	5 (1354 points)	Sonik "Blur"
4	1 (869 points)	Banal Projects "Seasick"
5	4 (772 points)	C-Lous "Assembly 95 40k Intro"
6	11 (714 points)	Extend "Sections"
7	9 (648 points)	Yodel "Oberon"
8	10 (427 points)	RRR "Cinderhatch"
9	13 (340 points)	Bomb Squad "Ultimate Stress"
10	6 (311 points)	Heretic "Delirium"
11	8 (272 points)	Royal "Sauna!"
12	15 (209 points)	Cirion "Hope"
13	14 (206 points)	Kinky "Sumu"
14	7 (188 points)	Amacon "Mission Impossible"
15	12 (108 points)	Diffusion "Kintro"

GRAPHICS

1	13 (1769 points)	Visualize/Jamm "Fiction"
2	6 (1457 points)	Artifec/Complex "Mystery"
3	7 (1285 points)	Jogi/Mellow Chips "Agony"
4	1 (1177 points)	Visigoth/Pure Resistance "Valkyria"
5	5 (948 points)	Kube/CNCD "Mustafa"
6	2 (880 points)	Kal/Astroidea "Morphosis"
7	12 (802 points)	Yoga/United Artists "An axe"
8	11 (702 points)	IronMan "Phobic"
9	4 (664 points)	Mazor/Paragon "Pain 2"
10	8 (639 points)	Facet&Super Nao/Lemon. "Baby"
11	10 (616 points)	Wolf/X-Trade "NO"
12	15 (428 points)	Tyshdomolo/Abyss "Grandma"
13	3 (372 points)	Zeb/Orange "Nut Fish"
14	14 (254 points)	Neutesten/Zirkonium "Blend"
15	9 (201 points)	Destop/CNCD "Wicked"

RAYTRACE

1	14 (2375 points)	Diffusion "The Desktop"
2	11 (1487 points)	Andy "Church Windows"
3	2 (1039 points)	Tapsah/Absolute Xtacy "Flower"
4	15 (981 points)	Dark Juha/Hirmu "Da End"
5	13 (976 points)	Toalnkor/Realtech "Sunset in Vectorcity"
6	10 (866 points)	Fish/Damane "Candle"
7	5 (737 points)	Marek Gibney "Jesu"
8	1 (466 points)	Spiff/Obsession Development "On"
9	4 (396 points)	Daemon/Dawn "Interface"
10	3 (392 points)	Minx/Fascination "Planscape"
11	6 (313 points)	Willysoft "Room of runes"
12	12 (263 points)	Serge "The Twilights"
13	7 (214 points)	- "World"
14	8 (191 points)	Havenlock/DKS "Grimreaper"
15	9 (147 points)	Turo/Fascination "Kielo"

ANIMATION

1	5 (2247 points)	Flow by Jaco
2	8 (2247 points)	Pulp by RRRR & Bang
3	11 (2156 points)	Space 01 by Cubic Team
4	10 (1346 points)	Chestmaster 2001 by Slimy Devil
5	2 (604 points)	Fastline in vector city by Toalnkor

6	6 (603 points)	Space Mountain by Execom
7	4 (384 points)	Dawn by Artifex
8	3 (307 points)	Bomb Ride by Vertex Twister
9	9 (175 points)	Insel by Marek Gibner
10	1 (147 points)	Cityscape by Zinx
11	13 (134 points)	The Choice by Jan-Erik Tervo
12	7 (118 points)	AAA-Animation by Shaq
13	12 (93 points)	NIH by Frank
14	14 (11 points)	Atlantis by Price and Tennessee
15	15 (6 points)	

4 CHANNEL

1	11 (1420 points)	Theseus/Anathema "Funkyeeh"
2	9 (1280 points)	Cube/Dee "Illumination"
3	2 (1193 points)	Lizardking/Razor 1911 "Crayfish Party"
4	7 (1080 points)	Breeze/Capacala "Aroma of northwest"
5	8 (1056 points)	Oxide/Sonik "Topless"
6	15 (921 points)	Ukulele/Banal Projects "Jormuan Lava"
7	6 (824 points)	Radix/Limited Edition "Rymdfunk"
8	12 (797 points)	Dizzy/CNCD "Suuntaviivat"
9	1 (742 points)	Brem/Bomb Squad "Get it"
10	4 (707 points)	Groo/CNCD "Pop Huora"
11	3 (579 points)	Andy/Banal Projects "Balthazar"
12	13 (462 points)	Boheme/Bomb Squad "The Robot Kingdom"
13	10 (458 points)	Sphinx/Fanatic "Holocaust"
14	5 (369 points)	Yoga/United Artists "Chuynia"
15	14 (330 points)	Dj. Roberto "Blowing House"

32 CHANNEL

1	7 (2999 points)	Skaven "Catch that Goblin"
2	5 (1017 points)	Rage "Guild of sounds"
3	11 (988 points)	Lizardking "World of unicorns"
4	2 (968 points)	Stargazer "Garage"
5	8 (886 points)	Prism "Can you feel it"
6	6 (811 points)	Beathawk "Breaking the sky"
7	4 (724 points)	Yolk "Temple of sun"
8	10 (699 points)	Purple Motion "Astraying Voyages"
9	9 (648 points)	Zake "Kaapu-Pete Matkustaa"
10	3 (528 points)	Pauli Rämö "Music for the forest"
11	1 (504 points)	Yzi "Spinning"
12	13 (380 points)	Turtle "Power Boogie"

13	12 (228 points)	RSE "So.if.fix.x."
14	14 (201 points)	Pete "Sy-Stems"
15	15 (200 points)	Edge "Farewell to Lyrics"

C64 DEMO

1	5 (3874 points)	Byterapers Inc. "Extremes"
2	4 (2227 points)	Panic "Break Through II"
3	6 (2212 points)	Beyond Force "7 Years"
4	3 (1132 points)	Crest "It's comming"
5	2 (1012 points)	Symptom C64 Section "Bizarre"
6	1 (647 points)	Chaos C64 Division "Tribute to Albert Hofmann"
7	7 (107 points)	-
8	8 (25 points)	-
9	12 (23 points)	-
10	14 (18 points)	-
11	9 (14 points)	-
12	11 (6 points)	-
13	15 (5 points)	-
14	13 (3 points)	-
15	10 (2 points)	-

C64 MUSIC

1	1 (2224 points)	Zardax/Origo Dreamline "Martinism"
2	3 (2016 points)	Thor/Extend "Kirta"
3	2 (1983 points)	Dr.Voice/Panic "Compomusic"
4	6 (1479 points)	AMJ/Side B "Sys 4096"
5	5 (996 points)	Warlock/Panic "Compotune"
6	4 (790 points)	Barracuda/Extend "Brainscanaloop"
7	7 (40 points)	-
8	12 (15 points)	-
9	8 (12 points)	-
10	9 (11 points)	-
11	10 (11 points)	-
12	15 (10 points)	-
13	11 (9 points)	-
14	13 (9 points)	-
15	14 (2 points)	-

C64 GFX

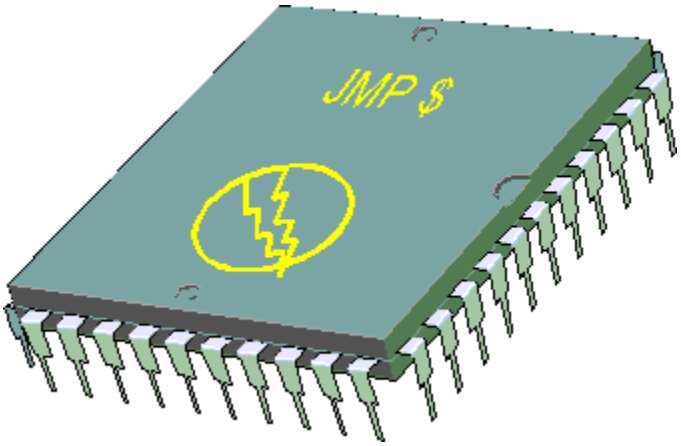
1	3 (2631 points)	Dr. Dick /Byterapers inc. "Dragon"
2	1 (2628 points)	Debris/Panic/Extacy "Compopicture"
3	2 (1636 points)	Votka/Pullo "Animaali"
4	6 (491 points)	
5	4 (335 points)	
6	5 (260 points)	
7	7 (37 points)	
8	10 (26 points)	
9	8 (24 points)	
10	12 (9 points)	
11	11 (5 points)	
12	9 (4 points)	
13	13 (3 points)	

PC 4K INTRO

1	8 (2100 points)	Animate
2	9 (1729 points)	Heaven
3	2 (1008 points)	Crashtest
4	13 (948 points)	Chrome
5	12 (836 points)	Havoc
6	5 (801 points)	Speed
7	10 (696 points)	Compexity
8	15 (608 points)	Dragon
9	4 (559 points)	Bit
10	11 (430 points)	Loop01
11	14 (366 points)	Ahrum
12	3 (340 points)	Only for few bytes
13	1 (337 points)	Redrum
14	6 (326 points)	Strictly 4kb
15	7 (324 points)	Shine

JMP $

Rixed [Jul 95]



En tant que codeur, il nous arrive par moment des aventures insolites, des tuiles inattendues ou plus rarement des miracles inespérés, qui échappent quotidiennement au quidam ordinaire. Il était temps de consigner ces faits héroïques.

Du fait, cette page aurait pu porter comme titre "Le coin des codeurs", mais je préfère "JMP_$", qui est d'une part plus original et d'autre part mon instruction préférée.

Au menu cette fois-ci : Comment avoir deux pages linéaires en mode 13h, un exemple d'IRQ verticale qui fonctionne, un petit truc à savoir à propos du codage des modes d'adressage, une pénalité facile à éviter pour 486, un éclaircissement au niveau des algorithmes de texture 3d, une petite gruge rigolote pour casser l'écran en plusieurs fenêtres, et pour finir un petit détour par le monde des virus.

Bon zapping...

La page cachée du mode 13h

Tout commença sur un petit board dans une région hostile en banlieue Nord. Là se tenait une conversation comme il s'en tient à peu près partout :

"- Ouais, la VGA c'est naze, on ne peut adresser qu'une seule page linéairement, entama une écriture.

- Oui, c'est bien dommage, répondit une seconde écriture (la mienne, quoique comme il s'agit d'ASCII, elle eut pu appartenir à n'importe qui sachant faire autant de fautes que moi), d'autant qu'il s'en faut de peu : regarde, en mode 13h on ne peut utiliser que 64Ko sur les 256Ko de la carte car les deux bits faibles de

toutes les adresses sont toujours remis à zéro. Imagine qu'un registre permette de choisir la valeur pour ces deux bits, et on pourrait écrire linéairement sur toute la VRAM, en changeant la valeur de ce registre pour écrire sur une page différente.

- Peut-être, reprit un troisième, mais alors il ne serait de toute façon pas possible d'afficher à l'écran une autre page que celle contenant les accès aux adresses divisibles par quatre, car comme chacun le sait, le compteur d'adresse du CRTC doit être décalé de deux bits vers la gauche (mode double-mot) pour pouvoir afficher un point sur quatre.

- Erreur, puisque le compteur d'adresse n'est pas décalé de deux bits mais *rotaté* de deux bits (Note aux CORs : rotater n'est pas un verbe très agréable à l'oreille mais il est en usage chez les codeurs, qui du reste se foutent de savoir s'il est agréable ou pas à leurs oreilles, puisque celles-ci ne sont sensibles qu'au son d'une GUS), et en mode double-mot les bits **14 et 15**se retrouvent à la position 0 et 1. Il serait donc possible d'afficher toute la VRAM en sautant 3 caractères sur 4, répondais-je pour achever mon raisonnement.

- Un caractère ? Mais on est en mode 13h pas en mode texte, banane, sentit opportun de placer un nabot qui passait par là (il y en a sur tous les boards de ces nabots-là, c'est la plaie) !

Je me permets de te rappeler, très cher, qu'un caractère, pour le CRTC, c'est un bloc de 4 points situés aux mêmes adresses sur chacun des quatre plans de bits. T'as pas lu le titre de l'article, il est question de codeurs, alors rentre vite à la maison car ta maman t'appelle, expédiais-je posément, tandis que j'envoyais ma santiag en direction du postérieur de l'intéressé, avec comme mission de lui laisser une marque tellement incrustée dans sa chair qu'elle se transmettra sur plusieurs générations, ce qui clût la conversation (Note aux CORs : hé, les mecs, je suis vraiment désolé pour le "clût", n'allez pas croire que je le fasse exprès mais d'après mon dictionnaire la forme passée de l'indicatif du verbe "clore" n'est pas usitée, j'ai donc dû inventer cette orthographe. Nécessité fait loi)

De retour chez moi, cette question continua de me trotter dans la tête. En partie parce qu'elle continuait de me trotter dans la tête, et en partie parce que je ne pouvais m'empêcher de faire le rapprochement avec le mode odd/even où il est effectivement possible de fixer la valeur du bit **0**de l'adresse d'un accès. Il s'agit du bit **5**du Misc à l'adresse **3c2** en écriture et **3cc** en lecture. Je consultais alors ma documentation secrète à propos de ce bit et ce que je vis me laissa perplexe comme un intégriste devant un manuel de cuisine : il y avait un second bit à coté de celui qui m'attirait, un second bit négligemment marqué "indéfini - ne pas tenir compte".

Mon oeil ! Il se pourrait bien que ce bit, associé avec l'autre, me permette de fixer la valeur des deux bits effacés en mode chain4 !

J'essayais aussitôt.

Juste après quoi mon cri de victoire retentit dans le quartier ("GAAAAA!!"). Il se passait bien quelque chose ! Mes accès en écriture n'aboutissaient plus au même endroit !

Pour voir où ils aboutissaient, je modifiais les bits **14**et **15**du compteur d'adresse du CRTC dans la ferveur et dans le registre **C**, et là, avec une joie que je ne cherchais plus à dissimuler, je vis apparaître les points précédemment pockés !

Malheureusement, il me fallut aussi renoncer au bit indéfini qui n'avait apparemment réellement aucune influence, et je compris qu'il faudrait me contenter de deux pages et non pas quatre.

Résumé de la situation :

Pour écrire dans la seconde page, mettre le bit **5**du Misc (**3c2**) à un. Pour écrire dans la première page, le remettre à zéro.

Pour afficher la seconde page, mettre le bit **15**du registre **C** du CRTC (**3d5**) à un. Le remettre à zéro pour afficher la première page.

J'en conçu aussitôt quatre petites routines :

```
AffPage1:
mov dx,3d4h
                        mov ax,0ch
                        out dx,ax
                        retn

                AffPage2:
                        mov dx,3d4h
                        mov ax,800ch
                        out dx,ax
                        retn

                SetPage1:
                        mov dx,3c2h
                        mov al,63h
                        out dx,al
                        retn

                SetPage2:
                        mov dx,3c2h
                        mov al,43h
                        out dx,al
                        retn
```

Fermement décidé à faire breveter cette trouvaille pour revendre par la suite les droits d'utilisation à une grosse firme, et devenir par la même billgathesement riche, je pris tout de même la précaution de vérifier mon programme de test sur d'autres machines.

A la fac tout d'abord. Ca ne marcha pas.

Puis chez un pote. Puis chez un deuxième. Puis un troisième. Ca ne marcha nulle part.

Puis sur tout le board !

Ah, ça fonctionna chez un autre user ! Enfin.

"- Quelle carte possèdes-tu, mon ami, l'assaillis-je

- Beuuuh, une carte orange zones 1 et 2, une carte d'abonnement à Glamou...

- Je voulais dire : quelle carte *vidéo* possèdes-tu, répondais-je, lèvres pincées, comme lorsque l'on vous répond que l'on possède une carte orange lorsque vous pensiez à une carte vidéo.

- Ah, et bien ma fois : une Trident"

Quelle carte est-ce que j'utilise moi ? Une Trident. Argh.

Aaaaaaaaaaaaargh !!!!

Ca ne marche exclusivement que sur les Tridents. Je suis vert. Essayons d'oublier. Pas facile.

L'IRQ verticale qui marche (parfois)

Comme s'en souviennent ceux qui ont lu mes notes de fin d'articles d'il y a quelques Reps, j'avais beaucoup cherché pourquoi l'IRQ verticale de la VGA ne fonctionnait pas (IRQ 9), avant d'aboutir à la conclusion qu'en fait si, elle marchait, mais il fallait tripoter un jumper sur la carte.

D'accord, mais une fois que vous aurez méprisement arraché le jumper fautif, vous ne serez peut-être pas beaucoup plus avancé, car la procédure à suivre pour diriger l'interruption n'est pas très facile à trouver, les diverses documentations restant pour le moins floues sur le sujet.

Pour vous éviter quelques heures de recherches, voici la marche à suivre :

/* Mettre le bit **4**du registre **11h** du CRTC à 1. /* Laisser le bit **5**à zéro (dans la bible il est écrit qu'il faut que ce bit soit sur 1 pour déclencher l'IRQ, mais c'est le contraire en fait)

/* Puis naturellement autoriser les IRQ **2** et **9** qui peuvent être masquées dans les deux 8259

Et voilà. Jusqu'ici ca va. Le problème c'est pour signaler à la VGA que l'interruption est terminée et qu'elle peut en renvoyer une autre quand elle veut. Pour cela il faut :

/* Lire l'ISR0 (port **3c2** en lecture) /* Remettre le bit **4**à **0**dans le registre **11h** du CRTC /* Aussitôt le remettre à 1

/* Bien sur ensuite il faut signaler l'EOI aux deux 8259

Voilà, maintenant à vous les joies des IRQ verticales !

Jamais d'index sans déplacement

Il nous arrive parfois d'avoir de désagréables surprises. Imaginez : vous codez une routine que vous optimisez aux petits oignons, car tout votre timing est basé sur elle. Vous assemblez, vous corrigez quelques bugs, vous lancez, et là... là... Et là, clignotements et saccades sont au rendez-vous.

Vous lancez votre débugger, armé jusqu'aux dents d'un flegme plein d'appréhension (ce qui, que les choses soient bien claires, ne signifie pas grand chose).

Et là, lorsque vous vous retrouvez face à votre code, les bras et/ou la mâchoire vous en tombent : Vos jolis **MOV [EDX*2],al** ont étés remplacés par de peu élégants **MOV [EDX*2+00000000h],al** qui prennent 4 octets de plus que prévus, un cycle de préchargement supplémentaire, qui font déborder votre inner loop du cache, bref, en un mot : célakatass ! "Mais pourquoiiii !?" hurlez-vous.

C'est la vie. Il n'existe pas de forme de codage qui permette de coder un index seul sans déplacement. Et lorsque vous appliquez un facteur à une base, ce n'est plus une base mais un index.

Pour corriger ce petit problème, deux solutions : utiliser une base nulle, genre **MOV [EDX*2+EBX],al**, ce qui est plus court et revient au même si **EBX** est nul, ou bien si le facteur est deux, faire plutôt **MOV [EDX+EDX],al**.

Dans le même genre, sachez également qu'il est impossible d'utiliser **EBP** comme base sans avoir en plus un déplacement d'au moins 8 bits, ce qui rajoute un octet a l'instruction.

Inutile de me répéter "Intel Suxx", il n'y avait aucune autre manière de faire, et si vous arrivez à faire mieux avec 3 octets d'OpCode pour coder toutes les instructions avec tous les modes d'adressage possibles, et bien, heu... vous n'y arriverez pas.

Une pénalité d'adressage 486

Vous savez tous comment fonctionne le pipe-line du 486 : les opérations aboutissant à l'exécution d'une instruction sont réalisées parallèlement. Pendant que l'on exécute la première instruction, on décode la seconde et on précharge la troisième, chacune de ces opération nécessitant un cycle d'horloge, ce qui fait que le temps pris par la plupart des instructions est d'un seul cycle (ceux qui pensaient à trois cycles lisent trop rapidement).

Or ce principe de pipe line n'est possible que si les instructions exécutées en premier n'ont pas d'influence sur le décodage des instructions suivantes. Si c'est la cas, le 486 stoppe le décodage de l'instruction suivante jusqu'à ce que l'instruction en cours d'exécution soit achevée, ce qui coûte un cycle.

Notez au passage que les 386 ne disposent pas de cette capacité à déterminer si une instruction aura des répercutions sur la suivante, ce qui fait que le processeur stoppe toujours le décodage d'une instruction pendant une exécution. C'est ce "semi-pipeline" qui fait que les instructions du 386 nécessitent généralement un cycle de plus que sur 486.

Bref, vous pouvez vous demander : "mais quelles instructions influent sur le décodage des suivantes ?" C'est très simple : les références mémoires ne peuvent pas être décodées à l'avance si l'on ne connaît pas la valeur des registres utilisés comme base ou index dans l'opérande.

Résultat des courses, si vous faîtes :

```
MOV BX,15
ADD AX,[BX]
```

Le 486 stoppera le décodage de l'ADD tant que le MOV ne sera pas exécuté, et vous perdrez un cycle.

Il convient donc d'éviter une telle situation en entrelaçant les instructions (pas pour tout le programme, pensez-y juste dans les boucles les plus internes).

Le calcul de la luminositée s'éclaircit

Chez les vikings codeurs, il est généralement admis que Gouraud ou Phong, tout est histoire d'interpolation, et que l'essentiel est d'écrire dans un coin de l'écran le nom de l'algorithme.

Il est donc fréquent de lire dans des news-groups orientés "coderz" quantités d'âneries qui font peter une durite aux codeurs graphistes sérieux qui tentent de faire comprendre à ces milliers de "coderz" qu'un Phong n'est pas un Gouraud et qu'un Gouraud n'est pas un Deep-shading. C'est pour préserver les dernières durites de ces codeurs regardant du patrimoine (dont je ne fais pas partie) que je vais énoncer ici (en priant pour ne pas faire d'erreur) les deux principes sus-nommés de Gouraud et de Phong.

Les deux méthodes ont pour but de lisser des surfaces polygonales 3d. Elles ne permettent pas de lisser les contours des objets qui paraîtront donc toujours taillées à la hache ou au démanteleur fibro-plasmique amorti, selon vos moyens.

Pour chaque point de l'objet, on connaît un vecteur normal. Je sais, il n'est pas très intelligent de parler de vecteur normal pour un point, mais vous voyez de quoi je parle : je parle d'un vecteur normal à la surface qui passerait par ce point si cette surface était continue d'une face à l'autre. Pour avoir ce vecteur, vous pouvez le rentrer manuellement dans vos données ou bien le calculer avec la moyenne des normales des faces environnantes ou bien avec un vecteurs qui va du centre de l'objet vers le point si votre objet est un patatoïde.

Bref, on dispose aussi du vecteur lumineux qui pointe de la source de lumière vers l'objet (vecteur qui n'est constant pour l'objet que dans l'hypothèse ou la source de lumière est à l'infini).

Le but du lissage est de donner à chaque point affiché une intensité proportionnelle au cosinus de l'angle entre le vecteur lumineux et le vecteur normal _EN_CE_POINT_.

Le problème, c'est qu'on ne connaît pas le vecteur normal en chaque point. Tout le principe de Phong réside dans le calcul d'une approximation raisonnable de ce vecteur à partir d'un nombre fini de vecteurs normaux, à chaque sommet d'un polygone. Gouraud lui se simplifie le travail en cherchant à "approximer" directement l'intensité finale.

Citons avant de commencer l'approximation la plus simple, celle qui consiste à reprendre le vecteur normal à la face pour tous les points de la face. Si l'on considère de plus que le vecteur lumineux est fixe sur la face (c'est à dire que la source de lumière est très éloignée), intensité pour toute la face est constante. Mais on ne lisse rien du tout.

Pour lisser, M. Gouraud à proposé (Quand ca ? beuuuuh... il y a quelques années quand même, hein...) d'interpoler linéairement intensité sur toute la surface du polygone. Si le polygone est un triangle, l'opération est très facile. Sinon, il faut veiller à ce que la face soit effectivement interpolable linéairement entre ces vecteurs aux sommets, mais bon bref, vivent les triangles.

Pour du Gouraud, on calcule donc intensité des points où l'on connaît la normale (produit scalaire) et on interpole ensuite cette intensité le long des côtés du polygone puis entre deux côtés pour remplir le polygone. On s'en tire donc avec une addition par point, avec un petit code gen du genre

```
i = 0
REPT MAX
MOV [DI+i],AH
ADD AX,BX
i = i+1
ENDM
```

On peut même faire mieux, mais là n'est pas le sujet.

Notez au passage, parce que ce n'est pas évident pour tout le monde (ca ne l'a pas été pour moi en tout cas), que l'incrément de couleur d'un point à un autre est constant sur tout le polygone.

L'intérêt du Gouraud est que la transition d'une face à l'autre se fait sans à-coup intensité (normal : deux polygones contigus partagerons les mêmes normales sur le côté commun). L'inconvénient c'est que les faces apparaissent toujours plates.

Pour remédier à ce problème, M. Phong proposa quelques temps après une amélioration à la méthode de Gouraud, qui consiste à interpoler les vecteurs normaux plutôt que intensité, et à RENORMER LES VECTEURS. Il est en effet nécessaire de garder la norme du vecteur normal constante si l'on veut donner l'impression de faire tourner ce vecteur, et donc au final d'avoir l'impression que la face plane est courbe.

Il est donc nécessaire de se faire un produit vectoriel entre le vecteur normal de chaque point et un vecteur lumineux qui pourra être lui aussi recalculé pour chaque point (si votre source de lumière n'est pas à l'infini).

Notez que si vous êtes prêt à calculer votre produit scalaire en coordonnées sphériques, c'est un jeu d'enfant d'interpoler le vecteur en coordonnées angulaires car alors il n'est plus nécessaire de "renormer".

L'intérêt du phong est que le rendu est exact. La lumière forme un halo sur une face qui ne parait plus plate.

Il existe bien sur des méthodes d'approximation qui permettent d'obtenir un effet similaire de "halo" sans le calcul point par point du produit scalaire entre deux vecteurs interpolés et renormés, mais aucune de ces méthodes, à ma connaissance, ne correspondent à un cas particulier de la méthode de Phong, il n'est donc effectivement pas convenable d'appeler ça du Phong.

D'autant que certaines de ces méthodes n'ont rien à envier question ingéniosité à celles exposés ci-dessus (qui ont quand même ce petit côté bête et méchant, il faut bien le dire).

But this story shall also be told...

Line Compare à répétition

Faut-il que je rappelle ce que c'est que le "Line Compare" ? C'est un compteur qui indique au CRTC la taille de la fenêtre avant la coupure d'écran. On trouve ca dans le registre **18h** du CRTC (plus deux autres bits qui se trouvent dans les registres **7** et **9**).

Ce que l'on ne sait pas généralement, c'est que ce compteur peut fonctionner plusieurs fois par image, ce qui permet de répéter plusieurs fois la même portion de VRAM à l'écran.

A quoi ca sert de répéter plusieurs fois la même chose ? Oh, pas à grand chose, mais c'est toujours utile par exemple pour des zooms hards à la "Good Bad & Ugly".

Nos amis les virus

Comment parler d'astuces marrantes sans toucher un petit mot des virus ?

Voici une petite arnaque à tomber par terre tellement qu'elle est bonne, tirée du virus V6000, dont le but est

de garder le virus en mémoire ... lorsque l'utilisateur reboote
à partir d'une disquette saine !

Si si, c'est possible, voici comment le virus s'y prend : Il écrit dans la CMOS qu'il n'y a pas de lecteur de disquettes (il faut recalculer le check sum pour que la gruge passe inaperçue, bien sur), et lorsque le micro reboote, le BIOS va inconditionnellement charger le secteur boot du disque dur (ce qui se fait en un rien de temps et ne laisse pas l'utilisateur se rendre compte de ce qui se passe), et dans ce secteur de boot le virus s'installe en mémoire, remodifie la CMOS pour réintégrer le lecteur de disquettes,puis simule un boot avec le BIOS. Au passage, il peut même, pour plus de sécurité, infecter la disquette (de toute façon, le virus est stealth).

Et hop, le tour est joué, l'utilisateur croit s'être débarrassé du virus alors qu'en fait pas du tout.

Le mot de la fin

Voici qui ravira tous les inadaptés qui se demandent encore ce que signifie le terme "multimedia", ou qui, comme Warrant, prétendaient avec cynisme que cela ne signifiait rien. Voici en effet le catalogue de la déRoute qui nous éclaire de sa magistrale définition :

" Un PC multimédia permet de travailler de façon interactive, de l'image fixe ou animée, du son et du texte. Il possède un lecteur CD ROM double ou quadruple vitesse, une carte son, des enceintes et une carte graphique suffisante pour une bonne qualité d'image". Qu'on se le dise.

Voila qui, on l'espère, incitera nos lecteurs a délaisser quelques temps la section 'lingerie' de ce catalogue.

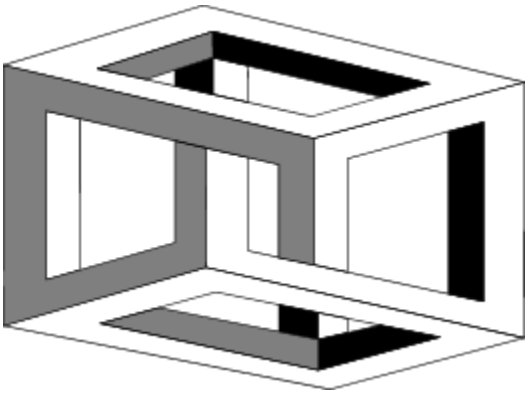
D'autant qu'on trouve aussi (page 1159) un indispensable glossaire que vous transporterez toujours avec vous lors des coding parties. Par exemple, vous serez peut être ravis d'apprendre qu'"Un 486DX convient parfaitement pour travailler sur tableur" (mais si vous voulez en plus jouer à PacMan, attendez le P6), que la RAM est cette chose qui "permet d'utiliser plusieurs logiciels en même temps", et autres merveilleusetées.

Bon, allez, assez plaisanté, on a tous du code qui nous attend.

A la prochaine.

La grande Saga de la 3d, épisode 3

Rixed



La grande Saga de la 3D - 3ème épisode

Assem Bhleur et le charme de Gouro

Tremblez, mortels ! Rixed le Chroniqueur va vous narrer les aventures du très terrible Chevalier Assem Bhleur au royaume du redoutable tyran Divaïd Hairor, il y a de cela très longtemps, in a galaxy far, far away.

Résumé des épisodes précédents

Après son terrible combat contre les redoutables soldats Cybers de la redoutable planète Interactivia de la redoutable galaxie Miltimidia (l'auteur entend améliorer le taux de compression Lempel-Ziff en utilisant toujours les mêmes adjectifs), qui s'est soldé par la retentissante déroute des imposteurs dont résonne encore la galaxie des lamentations de pitié et dont au sujet de laquelle il ne faut pas non plus oublier qu'il paraîtrait, mais il reste des sceptiques pour mettre en doute ces affirmations, qu'il ne faut jamais mélanger son KetchUp avec son Coca-Cola avant d'écrire un article.

Après bien des péripéties, le grand Chevalier Bhleur réussit toutefois à atterrir de justesse sur une planète résolument hostile, après il est vrai un soupir de soulagement. "Pfff, je ne mélangerai plus jamais mes aliments, pensa t-il".

Tout-à-coup, il souga/songi/songu/se mit à penser à sa fiancée retenue prisonnière loin d'ici par le terrible seigneur Divaïd Hairor. Soudain, le vizir apparut tout-à-coup et lui dit : " Chevalier Assem, ton courage est grand mais pour m'affronter, il te faudra réunir plusieurs items magiques, dont, pour commencer, et sans vouloir être désobligeant, le charme magique de Gouro qui transformera ton armure flat toute moche en une resplendissante armure +3 contre les runtimes errors."

"C'est vrai que mon armure manque de classe, se mis à songer Assem le preux Chevalier, et que je me verrais bien en dégradés de couleurs en fonction de la position de la source de lumière..."

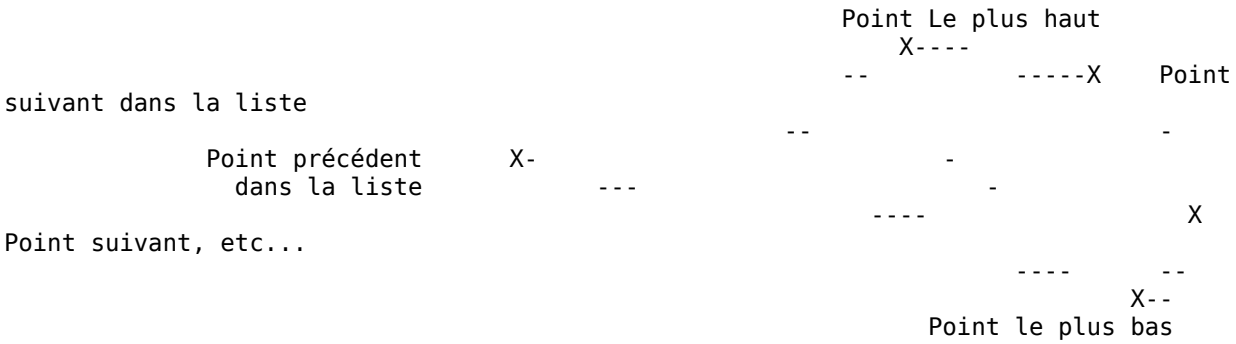
Souvenirs de Polygones flats

Avant de reprendre sa route, le chevalier considéra amèrement les plaques de son armure... Celles-ci étaient tracées à la va-vite, chacune d'une couleur distincte mais unique, comme vu dans le précédent épisode, mais comme revu plus en détail dans celui-ci car je ne suis pas payé pour raconter ces délires mais pour parler de 3d, la preuve en est, du tout, que je ne suis pas payé du reste. A moins que ce ne soit le contraire.

Quoiqu'il en soit, pour tracer chaque polygone à n côtés qui compose son armure resplendissante, mais, il est vrai, tristounette vis-à-vis de la comparaison qu'il a pu en faire avec les récentes armures des chevaliers disciples du grand Matt Moissa, sur la planète Demo II, il procédait de la sorte :

A partir de la liste des coordonnées 2D des n points composant la face, dans un ordre connu, disons l'ordre de ces antiques aiguilles des montres sans cristaux liquides, il recherchait tout d'abord le point le plus haut. Certains contestataires contestèrent que l'on pouvait tout aussi bien chercher le point le plus bas et faire tout le contraire de ce qui va suivre, ce qui revient au même, mais il est toutefois bon de garder à l'esprit le sort qu'aimait leur réserver Assem Bhleur, à savoir leur poquer des "JMP $" dans la ROM principale. Ceux qui ont subit cette exaction ne s'en vantent pas.

Une fois le point le plus haut trouvé, il est établi que le point à gauche de ce point à l'écran est celui qui le précède dans la liste, et le suivant celui qui le suit (certes), tout ceci moyenant un savant modulo des familles, dont un dessin explique tout aussi bien que ce charabia le divin fonctionnement :



(en m'excusant auprès du comité de soutient des dessins ASCII)

Pour se rendre d'un point à l'autre, il lui fallait se déplacer sur l'axe des X de (X2-X1)/(Y2-Y1) à chaque ligne, et ceci de chaque coté du polygone, afin de pouvoir tracer un segment horizontal de la couleur souhaitée entre deux points opposés du polygone...

Prenons l'exemple trivial du triangle :

liste de points :

```
NbrDePts            DW 3
Pt0                      DW 290,180                    ; X,Y
Pt1                      DW 20,110
Pt2                      DW 150,15
```

Vous remarquerez que les points sont effectivements dans l'ordre des aiguilles d'une montre.

Le programme remarque que le point le plus haut (Y minimal) est Pt2. Il initialise donc aussitôt un pointeur sur l'écran à l'adresse (150,15), adresse de Pt2, et il sait qu'il va falloir, en partant de ce point, atteindre d'un côté le point Pt0, et de l'autre Pt1. Il sait de plus que le point de droite sera Pt0 (après Pt2 - n'oubliez pas le modulo !), et le point sur la gauche Pt1, grace à l'ordre des points.

Il va donc calculer, pour la droite de gauche (et non pas la gauche de droite, ce qui, sorti d'un contexte strictement politique, ne signifie rien), l'incrément

$\delta Xg=(XPt1-XPt2)/(YPt1-YPt2)$,

incrément qu'il va prendre garde de multiplier par une puissance de 2, mettons 256, pour avoir une partie décimale (de 8 bits ici), en plus d'une partie entière. On a donc en fait :

$\delta Xg'=(256*(XPt1-XPt2))/(YPt1-YPt2)$

(remarquez qu'il avait bien pensé à multiplier par 256 AVANT de diviser, bien sur ! - n'est pas Chevalier Assem Bhleur qui veux).

Idem pour la ligne à droite, on a:

$\delta Xd'=(256*(XPt0-XPt2))/(YPt0-YPt2)$

Après avoir initialisé Xd et Xg à XPt2, il ne reste plus pour la routine qu'à boucler de YPt2 à Min(YPt0,YPt1) (=YPt1 ici), en faisant à chaque fois :

* Tracer le segment de Xg à Xd de la couleur du polygone
* Xg' += δXg'

Notez que Xg' est une variable dont la moitié haute est Xg et la moitié basse est la partie décimale de Xg. Par exemple, ici, puisqu'on a multiplié δXg par 256, on aura :

```
Xg'         LABEL WORD
               DB 0       ; la partie décimale
Xg           DB ?       ; la partie entière
```

* Xd' += δXd' (Même chose)
* Tant que Y < Min(YPt0,YPt1) GOTO Toulaho

Une fois arrivé à Min(YPt0,YPt1), il convient de recalculer le coeficient δX' concerné (sans toutefois modifier Xg', comme de bien entendu).

On voyait alors s'afficher les célèbres polygones flats qui avaient fait sa renomée, quelques anes et un paravent (oulala, bon courage !)

Mais, depuis ce temps, les frères perfides Inge et Nieur Duntel (planète Pintiôme, dite aussi "de la FIDIV mortelle") avaient méchamment accélérés leurs processeurs, et dorénavent sa flamboyante armure de jadis ne lui valait plus que les railleries et collibets des jeunes élèves du maître Cobol...

La mise au point du charme de Gouro

Tandis qu'il déambulait dans la jungle pleine d'affreuses grosses bêtes, Assem se demandait bien comment invoquer le charme du Gouro qu'il devait s'approprier avant de pouvoir quitter cette planète maudite entre toutes. Son armure en polygones flats, sans couleurs, lui faisait honte. Il n'oserait pas se présenter à la plus ringuarde des codings party de la galaxie avant d'avoir mis des couleurs dans sa texture...

Pour ce faire, il pensa tout d'abord à un vieux système qu'il avait jadis rencontré alors qu'il survolait une région primitive de la galaxie. La ruse était simple : tracer les segments de polygone d'une couleur qui était fonction de Y. Le résultat était de faire apparaître des dégradés de couleurs horizontaux. Mais cet artifice ne trompait plus personne depuis des générations de processeurs...

Le preux Chevalier Assem Bhleur s'assit au pied d'un arbre pour méditer, quand soudain la vue de la chute d'un oeuf trop mûr sortit son esprit de la torpeur qui s'installe toujours dans les esprits à ce moment de n'importe quel scénario de la trempe de celui-ci.

L'Oeuf était rond. Enfin, plus ou moins, mais nous allons supposer que si, sans quoi le récit s'en trouverait compliqué à un point que personne ne peut anticiper. Et donc, puisqu'il était rond, la lumière du soleil mutée de force dans ce coin de la galaxie se réflétait de manière diffuse sur sa coquille.

On eut dit que chaque rayon de soleil laissait à sa surface une lumière proportionnelle au produit scalaire entre le vecteur normal à la surface en ce point et ledit rayon lumineux, que, pour simplifier, nous supposerons venir de l'infini et donc être constant tout autour de l'oeuf ; ce qui, en considération de la taille de l'oeuf, de la taille du soleil, de sa distance avec la planète, et de l'heure tardive, est une approximation tout à fait acceptable.

Le vaillant Assem Bhleur se mit à imaginer ce que pourrait donner pareils reflets sur son armure : "Je vais octroyer à chaque point de mon armure (à chaque sommet des polygones, en somme), un vecteur normal à l'armure en ce point. Ceci me permettra de calculer la couleur de chaque point, en calculant le produit scalaire entre ce vecteur et les rayons lumineux incidents... Puis lorsque je tracerai les polygones, je n'aurai plus alors qu'à interpoler les couleurs entre chaque sommet des polygones, exactement comme je le faisais déjà pour les coordonnées X, pour avoir les couleurs du début et de fin de chaque segments à tracer. Il ne restera plus alors qu'à tracer les segments en interpolant une troisième fois entre ces deux couleurs !"

Chevalier Assem Bhleur venait à l'instant de reconstituer une version simplifiée du charme suprême du Gouro ! Il prononça alors le charme, "Interpolare Linearitatum Nostrum, Poilorum" et
la formule apparu :

δX'=(256*(XPt1-XPt0))/(YPt1-YPt0)

pour interpoler les coordonnées X de XPt0 à XPt1 (comme tout à l'heure, ca ne change pas)

δC'=(256*(CPt1-CPt0))/(YPt1-YPt0)

pour interpoler la couleur du bord des segments horizontaux entre la couleur CPt0 (Pour le point 0) et CPt1 (pour le point 1).

δCh'=(256*(Cd'-Cg'))/(Xg-Xd)

pour interpoler le long du segment horizontal, entre les couleurs Cg et Cd (couleurs du bord gauche et droit, respectivement) des points Xg et Xd (coordonnées X de gauche et de droite).

Ces interpolations se font toujours de la même manière en cumulant partie décimale et partie entière.

Cependant, Assem se rendit compte sans plus attendre que le coefficient était constant sur toute la surface du polygone. Et lorsqu'il se rendit compte que certaines personnes arrivaient à comprendre ce texte, il entreprit d'expliquer pourquoi : "En fait, ce polygone en deux dimensions peut en fait être considéré comme la projection orthogonale d'un polygone à trois dimensions sur l'écran, une troisième dimension représentant la couleur pour les points. Ainsi, le principe de l'interpolation linéaire montre que le triangle en 3d est plat (heureusement d'ailleurs sinon ce ne serait pas un vrai triangle de chez Triangles & co.). En conséquence, il est évident que lorsque l'on promène un vecteur sur sa surface la coordonnée Z (ou C, pour la couleur) reste constante avec les deux autres coordonnées. Et comme le triangle 2d est une projection plane, cela signifie ni plus ni moins que l'on peut se promener d'un vecteur constant dans le triangle 2d, quelque soit la direction de la promenade, en gardant un incrément constant pour la couleur, qui ne dépend que des coordonnées X et Y du vecteur-promenade."

Pour calculer l'incrément de couleur entre deux points situés consécutivement sur la même horizontale, Assem Blheur appliqua donc la formule :

ColorInc = (δC2-δC1)/(δX2-δX1).

Assem fit aussitôt une version plus sompteuse, en suivant ce vieil adage venu de la nuit des temps : "Plus y'a de normales, et plus c'est realiste !". Il songea donc qu'il n'était sans aucun doute plus nécessaire de considérer des polygones à n cotés mais uniquement des triangles, ceci afin d'améliorer le rendu ; car les triangles, sauf peut-être en plein coeur de trou-noirs spacio-interdimentionnels, là où diverses dimensions parallèles se rejoignent, se nouent et s'entortillent pour former un puissant rien du tout, et donc là où il est plutôt rare d'organiser des codings party, sont partout ailleurs les figures en géométrie 3d qui utilisent le moins de sommets (les objets sont donc plus complexes au sens où ils comportent plus de normales).

Confrontation avec Divaïd Hairor

C'est tandis que Chevalier Bhleur contemplait sa toute nouvelle armure que tonna le ciel et gronda le sol. Il faut dire qu'une porte spacio-temporelle venait de s'ouvrir devant lui, laissant la place à l'innégrotingnolant Divaïd Hairor, dont la félonie mérite bien à elle seule qu'on lui invente un adjectif réservé à son usage propre.

Divaïd Hairor se dressa un instant, laissant Assem estimer son équipement. Il portait une tenue de combat spacio-temporel mappée et shaddée à la perfection, sur laquelle on pouvait lire des greetz à profusion tournants à la demie-VBL.

Bhleur blémit, alors qu'il esquivait de justesse une quiche empoisonnée que venait de lui projetter Divaïd. Sa table d'interruptions fit aussi un soudain écart par décalage d'IDTR pour éviter la seconde attaque de Hairor, qui consistait en un peu subtil null pointer assigment.

Voyant toutes ses attaques contrées, Divaïd Hairor se résigna à employer l'un de ces plus terribles pouvoirs. D'un geste, il remplaça le processeur d'Assem Bhleur par un vieux 386sx16 !

Celui-ci, aussitôt, sentit sa nouvelle armure, dont il était si fier dix minutes auparavant, saccader sa démarche habituellement fluide.

Il fallait ruser. Assem Bhleur repensa tout à coup qu'il se faisait tard, et que depuis quelques lignes l'article n'avancait plus beaucoup. Alors, pour se sortir de ce pétrin, il imagina un subtil stratagème pour accélerer sa démarche chancelante.

Au lieu de tracer les segments horizontaux en interpolant la couleur dans une boucle, il coda plutôt une longue suite de :

```
MOV [DI-319],AH
ADD AX,BX
MOV [DI-318],AH
ADD AX,BX
MOV [DI-317],AH
ADD AX,BX
...
MOV [DI-3],AH
ADD AX,BX
MOV [DI-2],AH
ADD AX,BX
MOV [DI-1],AH
ADD AX,BX
MOV [DI],AH
ADD AX,BX
```

dans laquelle il sautait avec grace et délicatesse en fonction de la longueur du segment.

Ce subterfuge décupla sa vitesse, et Divaïd Hairor le terrible sultan ne put que se résigner à la défaite et à rendre au Chevalier sa dulciné. Compatissant, le Chevalier lui jetta : "Fuis, avant que je ne te détourne l'IRQ 0 sur l'INT 13h de ton beau SCSI".

Divaïd Hairor s'enfuit donc par la porte spacio-machinchose qui se referma derrière lui, tandis qu'Assem Bhleur rejoignit son formidable vaisseau pour quitter cette planète qui venait d'être témoin d'un des plus formidable afrontement que l'univers n'ai connu depuis le dernier épisode et en attendant le prochain.

The End

(but to be continued dans le Rep'9)
(ou 10)
(si je suis pas viré d'ici là)

Ne manquez pas le prochain épisode :

"Assem Blheur contre Docteur Fong"

[Mais que me veulent ces ambulanciers ?]

Programmation de la SB

TrIaX [Aou 95]

INTRODUCTION!!

Après avoir lu (et relu) les 6 premiers numéros du Rep', après avoir effectué ma prière quotidienne, la tête tournée vers le grand "ROR en chef", je me suis mis à réfléchir. J'ai réfléchi pendant, boh, allez, 5 secondes, et je suis parvenu à la conclusion que ma vie ne pouvait plus continuer comme avant. En effet, j'étais devenu comme esclave, esclave au service de mon maître qu'est le REP. Ainsi, me complaisant sans peine dans mon rôle, j'entrepris d'aller plus loin, et je commençai alors à écrire un article.

Un article, oui... mais SUR QUOI???

Ah ouais, c'est vrai, j'y avais pas pensé... Attends, attends, j'ai trouvé !!! Voilà enfin un sujet que le REP (journal fort complet au demeurant) n'a pas encore abordé : le SON !

Voilà en gros l'histoire de mes débuts et c'est ainsi que je vous présente l'article que j'ai eu tant de plaisir à écrire, en espérant que vous en aurez tout autant à le lire :

LA SOUND-BLASTER

programmation directe, i/o dma

OK, maintenant fini la plaisanterie, on commence à bosser.

Tout d'abord, il faut savoir que la SOUND-BLASTER (que je nommerai SB par la suite) est un immense standard, et qu'elle est (à ma connaissance) la première à avoir installé sur le marché du son un véritable outil de manipulation des digits.

C'est pourquoi de nombreux jeux (même parmi les plus anciens) reconnaissent et utilisent cette carte, et que l'on ne trouve quasiment aucun jeu récent qui ne gère pas cette carte.

Allez, allez, commence !!!!!

D'accord, d'accord. La SB possède quatre adresses de ports utiles à la gestion des digits :

```
                * 2x6h : port de RESET
(DSPReset)
                * 2xAh : port de lecture de données
(DSPRead)
                * 2xCh : port d'écriture/status du DSP                    (DSPWrite)
                * 2xEh : port d'état des données
(DSPDataState)
```

(DSP= Digital Signal Processor -je crois-, puce qui gère les digits)

Alors là, calme, le "x" n'est pas un nouveau chiffre hexa, c'est simplement un 'wildcard' qui sert à remplacer par le chiffre du port de base de votre SB (base=240h, x=4).

A part ça, je vois pas ce que je pourrais ajouter d'autre, je pense que c'est suffisament clair.

Maintenant qu'on a vu à quelles portes frapper pour réveiller le DSP, on va étudier comment taper un brin de causette avec la petite pupuce, à savoir comment réaliser des entrées/sorties de données sur le DSP.

1/LECTURE :

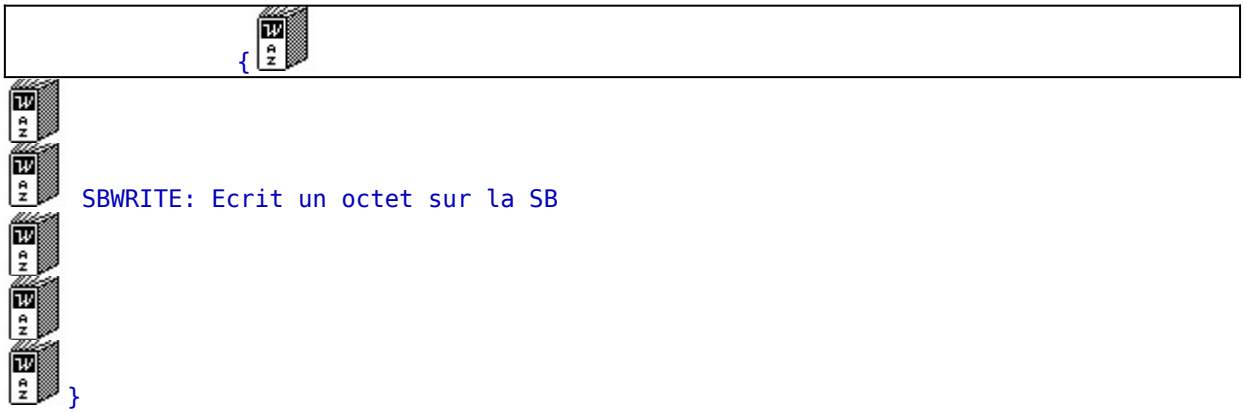
Pour lire une donnée sur le DSP, il faut d'abord attendre qu'il y ait quelque chose à lire, et pour cela il faut boucler jusqu'à ce que le bit 7 du port 2xEh passe à 1. A ce moment-là, il est possible de lire la donnée sur le port 2xAh.

```
              {

 SBREAD: Lit un octet sur la SB


}
               Function SBRead:BYTE; assembler;
               Asm
                              mov          dx,[SBPort]
                              or            dl,DSPDataState
                   @SBRead_Wait:
                                  in            al,dx
```

```
                    or          al,al
                    jns         @SBRead_Wait
                and         dl,0f0h
                or          dl,DSPRead
                in          al,dx
        End;
```

2/ECRITURE :

Là pareil, pour écrire un octet, il faut attendre... que le bit 7 du port 2xCh passe à 0, et ensuite écrire la donnée au même port 2xCh.




```
        Procedure SBWrite(Data:BYTE); assembler;
        Asm
                mov         dx,[SBPort]
                or          dl,DSPWrite
            @SBWrite_Wait:
                    in          al,dx
                    or          al,al
                    js          @SBWrite_Wait
                mov         al,[Data]
                out         dx,al
        End;
```

Jusque là, tout va bien ? Oui, bon alors je continue.. On va maintenant s'attaquer à un truc encore plus marrant et *wow* top délire, le RESET et puis tant qu'on y est l'AUTODETECTION de la SB.

```
                    Pour effectuer un reset du DSP, il faut :

                    * écrire 1 au port de reset (2x6h)
                    * attendre 3 microsecondes (quelques INs)
                    * écrire 0 au port de reset
                    * lire un octet
                    * si l'octet=0AAh, c'est gagné!!
```

Deux cas se présentent alors :

- Le premier, simple, on ne trouve pas à la fin l'octet 0AAh. Alors là, catastrophe, le reset n'a pas marché.

- Deuxièmement, si on tente un reset sur un machine qui ne possède pas de SB, le test pour la lecture de l'octet (souvenez-vous, c'était il y a quelques lignes) ne risque surement pas de marcher et youplaboum on se retrouve dans une boucle sans fin. D'où la nécessité d'introduire un compteur limitant le nombre maximum de bouclages (utile uniquement lors du reset).

```
{
 SBRESET: Balance un reset au DSP

}
                Function SBReset:BOOLEAN; assembler;
                Asm
                            {

 Envoie le reset

}
                                mov         dx,[SBPort]
                                or           dl,DSPReset
                                mov         al,1
                                out         dx,al
                                    in          al,dx
                                    in          al,dx
                                    in          al,dx
                                    in          al,dx
                                xor         al,al
                                out         dx,al

                            {

 Attend une réponse de la SB
```

```
}
                                and          dl,0f0h
                                or            dl,DSPDataState
                                mov          cx,256
                        @SBReset_Wait:
                                   in           al,dx
                                   or           al,al
                                   js           @SBReset_Test
                                   loop        @SBReset_Wait    {<-- C'est ça le
compteur!}
                        @SBReset_Failed:
                                   mov          al,FALSE
                                   ret

                            {
```

```
  Teste si on a bien affaire à un DSP
```

```
}
                    @SBReset_Test:
                                and          dl,0f0h
                                or            dl,DSPRead
                                in            al,dx
                                cmp          al,0aah
                                jne          @SBReset_Failed
                                mov          al,TRUE
                End;
```

Poussons la réflexion un peu plus loin.

Avec cette routine, on est devenu capable de tester si une SB est présente ou non à un port précis. Ca vous donne pas des idées ?? Non ?? Bon bah tant pis.

Euh non, je vais vous dire à quoi je pense : à partir du moment où l'on sait que les cartes SB peuvent être configurées sur des adresses allant de 210h à 280h, il suffit de tester chacun des ports un par un pour trouver l'adresse de base de la SB installée (si il y en a une installée, bien sûr !!).

Voilà la routine :

```
                {
```

```
  SBDetectPORT: Détecte le port de base SB
```


}

```
Function SBDetectPort:BOOLEAN;
Begin
      SBDetectPort:=FALSE;
      SBPort:=$210;
      Repeat
            If SBReset then
                Begin
                      SBDetectPort:=TRUE;
                      Exit;
                End;
            Inc(SBPort,$10);
      Until SBPort>$280;
End;
```

Le DSP est reseté, on a trouvé l'adresse de base, c'est tout cool ! Sauf que maintenant, ça devient casse-bonbons... On va commencer à faire du sérieux : envoyer des commandes au DSP.

Une commande, c'est tout simplement un octet que l'on envoie au DSP, suivi de un ou plusieurs ou pas d'arguments, suivant la commande. Voici une liste de quelques-unes des commandes du DSP :

```
* 10h = Direct sample output                              (+sample)
* 20h = Direct sample input                               (renvoie
sample lu)
* D1h = Set HP on
(pas d'args)
* D3h = Set HP off
(pas d'args)
* E1h = DSP version number                                (voir plus
bas)
```

Quelques petits commentaires :

Les samples sont au format non signé, c'est à dire que les données s'étendent de 0 à 255 (et non pas -128 à 127 comme sur Amiga ou bien GUS).

Pour la commande E1h, si la version de votre DSP est 2.01, la bêbête renverra d'abord 2, puis 1.

Quelques petits exemples :

```
{
```

```
SBDirectIN: Lit un sample direct from SB
}
            Function SBDirectIn:BYTE;
            Begin
                  SBWrite($20);
                  SBDirectIn:=SBRead;
            End;

            {
```

```
SBDirectOut: Sort un sample direct to SB
}
            Procedure SBDirectOut(Sample:BYTE);
            Begin
                  SBWrite($10);
                  SBWrite(Sample);
            End;
```

Je vous laisse le soin de coder les autres, ou bien de simplement jeter un coup d'oeil sur le fichier SB.PAS pour une implémentation complète de toutes les routines.

Voilà, mon article est fini, vous pouvez aller vous coucher... Allez,
bonne nuit!!

(c)TrIaX'95!

Quoi ?? Vous en voulez encore ????

Vous avez raison, j'ai gardé le meilleur pour la fin :

La Sound-Blaster

Le Transfert D.M.A.

Avant d'expliquer l'i/o dma par la SB (très simple), on va d'abord s'intéresser à ce qui se passe du côté du composant DMA.

LE DMA - COMMENÇONS PAR LA THÉORIE.

L'initialisation d'un transfert DMA requiert 7 étapes :

```
1) 'Disabler' le canal DMA
2) Remettre la bascule (Flip-Flop) DMA à 0
3) Sélectionner le mode de transfert
4) Envoyer la longueur du bloc à transférer
5) Envoyer l'addresse du bloc à transférer
6) Envoyer la page du bloc à transférer
7) 'Enabler' le canal DMA
```

Alors là, tout homme (ou femme car, on ne le dira jamais assez, les femmes sont autant présentes que les hommes dans l'univers "interactif-multimédia-intel inside" dans lequel nous vivons), je disais donc tout homme (ou femme) normalement constitué se dit : ATCHOUBIDOU ?!?

N'est-ce pas ?

Mais... Que se passe-t-il ? Quel est ce bruit courant dans le lointain ? Mais... Mais oui ! C'est lui ! C'est TrIaX ! sur son beau cheval blanc, qui vient nous éclairer sur les énigmes qu'il nous a laissées !

Exactement. Procédons par ordre :

- **'Disabler le canal DMA'** :
  cela indique au composant DMA qu'il doit laisser libre ce canal pour pouvoir être reprogrammé.

- **'Remettre le Flip-Flop à 0'** :
  étant donné que la longueur et l'adresse sont des données 16bits, et et que le port par lequel sont envoyées ces données ne contient que 8bits, y'a un truc ! Et ce truc c'est le Flip-Flop : on envoie d'abord l'octet Lo, puis Hi.
  -> Remettre à 0 le Flip-Flop consiste à forcer la bascule sur l'attente
  d'un octet Lo.

- **'Sélectionner le mode detransfert'** :
  indique au DMA quel type de transfert nous allons effectuer : transfert en lecture/écriture, incrémentation / decrémentation des adresses... (voir plus loin)

- **'Envoyer la longueur du bloc'**:
  ben, c'est simple ? On envoie la longueur du bloc de données à transférer en envoyant d'abord l'octet Lo du mot, puis l'octet Hi.

- **'Envoyer l'addresse du bloc'** :
et - **'Envoyer la page du bloc'** :

Alors là, ça se corse un peu. L'adresse et la page du bloc sont calculées à partir de l'addesse physique du bloc calculée à partir du couple Segment:Offset de toute addresse 80x86 (?!)

Les formules sont donc :

```
Addresse_Physique = Segment * 16 + Offset
Page                          = Addresse_Physique shr 16
Addresse                   = Addresse_Physique and 0ffffh
```

Autrement dit, la page représente les 4 bits de poids fort de l'adresse physique, tandis que l'adresse du bloc représente les 16 bits de poids faible.

- Enfin **'Enabler le canal DMA'** :
C'est l'opération inverse du 1), cela indique au DMA de laisser libre le canal aux périphériques qui peuvent alors l'utiliser pour le transfert.

LE DMA - LA PRATIQUE, Y'A QU'çA D'VRAI ! (tiens,tiens...)

En effet.

Je vous le dit carrément : là, dans votre PC, il n'y a pas UN mais DEUX composants DMA... Mais oui, mais oui, vous l'auriez jamais cru, hein ? Mais pourtant c'est vrai. Deux composants DMA. Un 8bits, et un 16bits. Alors là, vous me dites: "ah ouais, d'accord, c'était pas la peine de faire le malin !". Mais justement, si. Parce que figurez vous que ces deux composants sont situés à des adresses différentes et, étant donné la profusion d'adresses liées à ce composant, hé bien : profusion x2 ça fait quand même beaucoup.

Allez, n'ayez pas peur, je vous file un tableau récapitulatif :

	Composant DMA n°1							Composant DMA n°2	
Canal DMA	*00*	*01*	*02*	*03*	*04*	*05*	*06*	*07*
Enable/Disable	00Ah	00Ah	00Ah	00Ah	0D4h	0D4h	0D4h	0D4h
Mode transfert	00Bh	00Bh	00Bh	00Bh	0D6h	0D6h	0D6h	0D6h
Flip-Flop	00Ch	00Ch	00Ch	00Ch	0D8h	0D8h	0D8h	0D8h
Longueur bloc	001h	003h	005h	007h	0C2h	0C6h	0CAh	0CEh
Addresse bloc	000h	002h	004h	006h	0C0h	0C4h	0C8h	0CCh
Page bloc	087h	083h	081h	082h	08Fh	08Bh	089h	08Ah

Et maintenant voici la structure des registres :

```
            * Enable/Disable : bits 0-1 = (n° Canal) and 03h
                                                bit    2      = 0-> Enable
                                                                1-> Disable
                                                bits 3-7 = 'reserved'

            * Mode :                        bits 0-1 = (n° Canal) and 03h
                                                bits 2-3 = 00-> transfert verify
                                                                01->
transfert en écriture
                                                                10->
transfert en lecture
                                                bit 4        = 0-> Disable Auto-
init
                                                                1-> Enable
Auto-init
                                                bit 5        = 0-> Address
increment    (à l'endroit)
                                                                    = 1-> Address
decrement    (à l'envers)
                                                bits 6-7 = 00-> demand mode
                                                                01-> single
mode
                                                                10-> block
mode
                                                                11-> cascade
(je sais pas du tout à quoi
correspondent    ces bidules
mais    je    sais    qu'on doit
utiliser 'single mode'...)
```

En fait, ne vous embarrassez pas avec toutes ces précisions, nous utiliserons nous le mode 44h lors d'un transfert en écriture et le mode 48h lors d'un transfert en lecture.

Voilà, j'ai tout dit. (En fait, tout ce que je savais, car ma référence sur le DMA n'est pas du tout complète, je lance donc un appel aux coders de tous poils pour m'aider ou corriger les fautes que j'ai pu commettre dans ce chapitre).

Je vous donne donc la routine d'init DMA:

```
      Const
        DMAMask          : array[0..7] of byte=($0A,$0A,$0A,$0A,$D4,$D4,$D4,$D4);
        DMAMode          : array[0..7] of byte=($0B,$0B,$0B,$0B,$D6,$D6,$D6,$D6);
        DMAFlipFlop : array[0..7] of byte=($0C,$0C,$0C,$0C,$D8,$D8,$D8,$D8);
        DMACount        : array[0..7] of byte=($01,$03,$05,$07,$C2,$C6,$CA,$CE);
        DMAAddress    : array[0..7] of byte=($00,$02,$04,$06,$C0,$C4,$C8,$CC);
        DMAPage          : array[0..7] of byte=($87,$83,$81,$82,$8F,$8B,$89,$8A);

      Procedure DMAInit(Channel:BYTE;Mode:BYTE;Count:WORD;Address:POINTER);
      assembler; asm
                  xor          dx,dx
•
                  xor          bx,bx
                  mov          bl,Channel

                {
```

```
  Disable canal DMA
```

```
  }
                  mov          dl,[offset DMAMask+bx]
                  mov          al,bl
                  and          al,00000011b      {Pas vraiment utile car al toujours <
8}
                  or            al,00000100b
                  out          dx,al

                {
```

```
  Remet Flip-Flop à 0
```

```
  }
                  mov          dl,[offset DMAFlipFlop+bx]
                  xor          al,al
                  out          dx,al

                {
```

```
  Sélectionne le mode de transfert
```

```
  }
                  mov          dl,[offset DMAMode+bx]
                  mov          al,bl
                  and          al,00000011b
```

```
                or            al,Mode
•
                out           dx,al
•
            {
```

Envoie la taille du bloc à transférer

```
            }
                mov           dl,[offset DMACount+bx]
                mov           ax,Count
                out           dx,al
                mov           al,ah
                out           dx,al

            {
```

Envoie l'adresse et la page du bloc

```
            }
                mov           cl,4
                mov           ax,word ptr [Address] {AX=Seg}
                mov           ch,ah
                shr           ch,cl
                shl           ax,cl
                add           ax,word ptr [Address+2] {Offset}
                adc           ch,0

                {To the dma...}
                mov           dl,[offset DMAAddress+bx]
                out           dx,al
                mov           al,ah
                out           dx,al
                mov           dl,[offset DMAPage+bx]
                mov           al,ch
                out           dx,al

            {
```

Enable canal DMA

```
            }
                mov           dl,[offset DMAMask+bx]
```

```
                mov         al,bl
                and         al,00000011b
                out         dx,al
    End;
```

REVENONS A LA SB

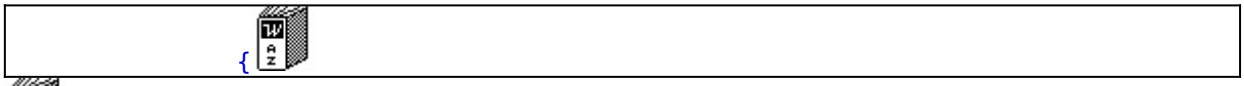
Oui, c'est vrai, ne nous écartons pas du sujet.

Le transfert via DMA est beaucoup plus simple du point de vue de la SB. On utilise pour ce faire les commandes 14h pour sortir du son et 24h pour en sampler, après avoir bien entendu pris soin de régler la fréquence de sampling par la commande 40h.

Ca s'passe comme ça (lalala...) :

```
                    1) Régler la fréquence
                          * Envoi de la commande 40h
                          * Envoi de la constante pour le timer interne de la SB
                              qui se calcule comme suit :

                              Timer_Constant = 256 - 1000000 / Fréquence en Hz


                    2) Demander un transfert DMA
                          * Initialisation du DMA
                          * Envoi de la commande (14h ou 24h, c'est selon..)
                          * Envoi de la longueur du transfert-1
                              D'abord l'octet Lo, puis l'octet Hi.
```

Et voilà les routines :



```
{

SBSetFreq: Règle la fréquence de sampling

}
                Procedure SBSetFreq(Freq:WORD);
                Begin
```

```
                SBWrite($40);
                SBWrite(256-1000000 div Freq);
        End;

        {
```

```
  SBDmaIn: Enregistre un sample via DMA
```



```
}
        Procedure SBDmaIn(Count:WORD;Address:POINTER);
        Begin
                DMAInit(SBDma,DMARead,Count,Address);
                Dec(Count);
                SBDmaDone:=FALSE;
                SBWrite($24);
                SBWrite(Lo(Count));
                SBWrite(Hi(Count));
        End;

        {
```

```
  SBDmaOut: Sort un sample via DMA
```



```
}
        Procedure SBDmaOut(Count:WORD;Address:POINTER);
        Begin
                DMAInit(SBDma,DMAWrite,Count,Address);
                Dec(Count);
                SBDmaDone:=FALSE;
                SBWrite($14);
                SBWrite(Lo(Count));
                SBWrite(Hi(Count));
        End;
```

Et c'est pas fini. Il reste encore 2 commandes concernant les transferts DMA, j'ai nommé : DMA Continue & DMA Stop. Bon, les noms sont suffisamment explicites, on va pas en faire un fromage, je vais juste vous donner les codes de ces commandes sinon vous seriez bien avançés :

```
                - DMA Continue    : 0D4h
                - DMA Stop           : 0D0h        (et    ces    deux commandes ne
requièrent
                                                                        aucun
paramètres ni l'une, ni l'autre)
```

Je ne vous file pas le code parce que :

1/ ça commence à bien faire;
2/ c'est suffisament simple; et
3/ y'a pas d'raison pour que ce soit toujours moi qui bosse...

M'enfin (vive GASTON !), il reste tout de même un truc à vous expliquer avant que vous alliez vous ruer comme des bêtes sur le code que j'ai si soigneusement et si amoureusement (et si longuement, n'est-ce pas SQUIRREL ?) préparé !

UN DERNIER TRUC AVANT LA FIN ;-(

Et oui, c'est pas encore fini ! (Mais quand s'arrêtera-t-il ?)

Bon, après avoir si gentillement commandé un transfert DMA à notre chère SoundBlaster, et après que celle-ci nous aie si aimablement restitué le son au travers de ses petits haut-parleurs (allumés j'espère, sinon ça marche pas...) et bien son travail (à la SB) n'est pas encore tout à fait terminé car elle nous envoie alors une IRQ.

```
                        IIIIIIII    RRRRRRR        QQQQQQ            ?????
  ?????
                              II          RR        RR    QQ        QQ        ??
??    ??      ??
                              II          RRRRRRR      QQ    Q QQ               ??
??
                              II          RR    RR        QQ      QQ
  ??              ??
                        IIIIIIII    RR      RR        QQQQQ QQ

??              ??
```

Cékoiçà ? (Mais non, je ne vous prend pas pour des débiles mentaux ...)

J'explique rapidement : une IRQ, c'est une INT (interruption) genérée par le matériel (en l'occurence notre SB). Et les irqs se situent à des vecteurs d'interruptions bien définis : de l'int 08h à 0Fh pour les irq 00h-07h, et de l'int 70h à 77h pour les irq 08h-0Fh.

Pour finir, deux choses à ne pas oublier de plus en plus en sus :

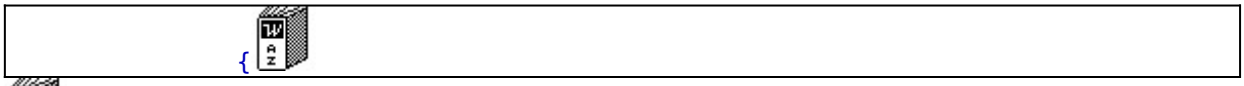
1-> Démasquer l'irq par l'intermédiaire du PIC (Programmable Interrupt Controller), à savoir envoyer un octet avec le bit correspondant au numéro de l'irq (ou (irq and 07h) pour les irqs>=8) positionné sur 0 au port 021h si l'irq<8 sinon au port
0A1h pour les autres.

2-> Dans le handler d'irq (la routine appelée lorsque l'irq survient), ne pas oublier d'envoyer une commande

d'acquittement d'irq (020h) au port 020h si l'irq<8 ou bien au port 020h *puis* 0A0h si irq>=8.

Du côté de la SB : lire un octet à son port DSPDataState, je sais pas pourquoi, mais c'est marqué dans toutes les docs et c'est dans toutes les routines que j'ai pu voir...

Je vous donne enfin les routines qui font tout ce boulot :



```
{
SBIrqHandler: Handler d'irq
}
              Procedure SBIrqHandler; interrupt; assembler;
              Asm
                              mov          [SBDmaDone],TRUE    {C'est en fait la
fonction de

cette routine, prévenir de la

fin d'une IRQ...}

                              mov          dx,[SBPort]            {Sert à que dalle}
                              or            dl,DSPDataState    {mais c'est dans la
doc...}
                              in            al,dx

                              mov          al,20h                 {Commande
acquittement d'irq}
                              cmp          [SBIrq],8
                              jb            @SBLowIrq
                              out          0a0h,al
                  @SBLowIrq:
                              out          020h,al
              End;

              {
SBInit: Installe le handler d'irq
```

```
}
                Procedure SBInit;
                Begin
                      If SBIrq<8 then
                          Begin
                                GetIntVec(SBIrq+$8,OldHandlerAddr);
                                SetIntVec(SBIrq+$8,@SBIrqHandler);
                                Port[$21]:=Port[$21] and (not (1 shl SBIrq));
                          End
                      Else
                          Begin
                                GetIntVec(SBIrq+$70,OldHandlerAddr);
                                SetIntVec(SBIrq+$70,@SBIrqHandler);
                                Port[$A1]:=Port[$A1] and (not (1 shl (SBIrq and 7)));
                          End;
                End;

                {
```

```
SBDone: UnInstalle le handler d'irq
```

```
}
                Procedure SBDone;
                Begin
                      If SBIrq<8 then SetIntVec(SBIrq+$8,OldHandlerAddr)
                                      else SetIntVec(SBIrq+$70,OldHandlerAddr);
                End;
```

Et en prime une 'tite routine pour détecter l'irq (construite sur le même principe que celle de l'autodétection du port) :

```
                {
```

```
SBDetectIRQ: Détecte l'irq SB
```


}

```
Function SBDetectIrq:BOOLEAN;
Const Irqs : array[0..4] of byte = (2,3,5,7,10);
Var
    IrqIndex : byte;
    IrqDelay : word;
Begin
      SBDetectIrq:=FALSE;
      SBHpOff;

      {
```

Parcourt toutes les irqs possibles à travers IrqIndex


}

```
      IrqIndex:=0;
      Repeat
            SBIrq:=Irqs[IrqIndex];
            SBInit;    {Installe handler}
            SBDmaOut($0002,Ptr($0000,$0000));    {Envoie un chti bloc}

            {
```

Attend pour voir si la SB répond


}

```
            For IrqDelay:=0 to $ffff do begin
                If SBDmaDone then
                    Begin
                          SBDetectIrq:=TRUE;    {
```

Oui-> cool on a trouvé!

}

```
                          Exit;
                    End;
            End;

            {
```

Non-> alors irq suivante

}

```
            SBDone; {Uninstall handler}
            Inc(IrqIndex);
      Until IrqIndex > 4;
End;
```

Et voilà, maintenant c'est vraiment terminé... Bah, allez, faut pas s'en faire, un jour viendra, où nous nous retrouverons, où nous nous parlerons de nouveau, moi jouant les professeurs, et toi m'écoutant, les yeux ronds et brillants, ébahi...

MAIS ARRETE DE DELIRER !!!!! (et conclue vite fait..)

Ok. Bon en fait j'ai plus rien à dire, c'était juste pour passer les 650 lignes... Sur ce, je vous donne le bonjour et peut-être rendez-vous pour une prochaine aventure... (qui sait ?)

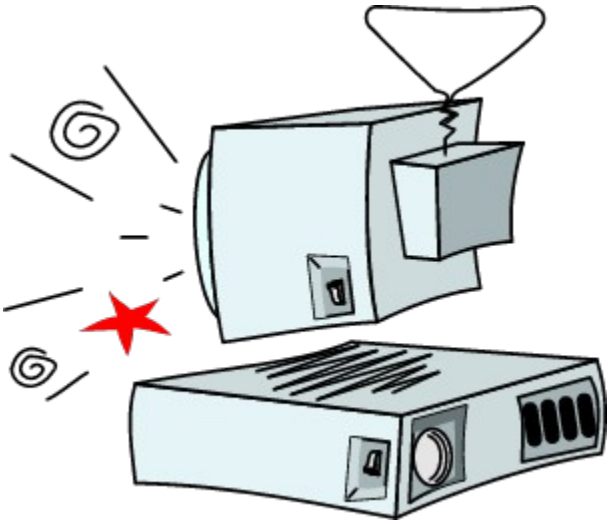
zissize (c) TrIaX!'95

(psss.. lancez SBTEST.EXE, c'est marrant !)

(Programme en annexes, Attention ! Carte SB et microphone requis !)

RunTime Error

Squirrel [Oct 95]



Présentation du DiskMag RunTime Error

Comme toujours, le Reporter est heureux d'assister à la naissance de fanzine et vous en présente un très prometteur : Runtime Error.

RunTime Error est un diskmag doté d'une interface très conviviale intégrant images, sons, musiques et animations. Et tout ceci accessible à tous les PCistes DOSmaniacs que vous êtes !

Vous pourrez trouver entre autres dans le fanzine :

- **Tests de jeux**, accompagnés de bruitages, images et fiches détaillées ;
- **Trucs & astuces**, bidouilles, tuteur pour de nombreux jeux ;
- **Articles**, Programmation, assembleur, turbo pascal, etc...
- Des **sharewares/freewares**créés par les auteurs !

Mais surtout : une bonne ambiance, où les articles se veulent accessibles à tous, du débutant au programmeur confirmé.

Le fanzine en est déjà à son troisième numéro, et ne cesse de s'améliorer, il devrait continuer sans problèmes sa rapide progression à l'aide de vos nombreuses participations !

Les créateurs de Runtime Error cherchent aussi des développeurs, auteurs, graphistes et musiciens pour enrichir les numéros suivants.

Pour obtenir les numéros 1, 2 ou le 3 qui va bientôt paraitre, proposer vos talents d'auteur, compositeur ou musicien écrivez à :

Loll :
Christophe Laurent
1, place du marché
67100 Strasbourg

ou

Tataye :
Schaeffer Nathanaël
6A, place d'Austerlitz
67000 Strasbourg

N'oubliez pas de joindre une disquette par numéro et une enveloppe timbrée et libellée à votre adresse.

Vous pouvez aussi contacter les auteurs sur **3614 Teaser**ou sur le **Bbs Music Power au (16) 88.83.63.99**en b.à.l Loll et Tataye.

Initiation au Qbasic IV

Squirrel [Sep 95]

Cette fois ci, vous n'aurez droit qu'à un mini article, mais rassurez vous, Qb5 sera plus fourni.

 Dessin en mode graphique

 Jouer de la musique

 Votre programme

 Errata Qbasic 3

Mode graphique

Squirrel [Sep 95]

I. Mode graphique

Les pc récents (comptez à partir du 386) sont généralement pourvus d'un écran VGA. Cet écran VGA possède deux modes bien distincts qu'il ne faut donc pas confondre :

- Le mode texte, celui dans lequel nous avons travaillé jusque ici qui n'affiche que des caractères bien définis (C'est celui dans lequel vous êtes dans le Reporter DOS).

- Le mode graphique, celui dans lequel nous ALLONS travailler pour la durée de cet article. Dans le mode graphique, on n'affiche pas des caractères, mais uniquement des points. A partir de ces points, une multitudes d'autres figures peuvent être obtenues : Lignes, arcs, cercles, rectangles et même de petites animations sont envisageables. Il est également possible en mode graphique d'afficher des caractères, mais ceux ci sont bien sûr eux aussi composés à partir de points. A titre de comparaison, Le Reporter Windows, lui, est en mode graphique.

Il existe une floppée de modes graphiques (tout comme les modes textes d'ailleurs, mais ne nous égarons pas...).

Un mode graphique est défini par :

- Sa résolution horizontale (nombre de points dans les X),
- Sa résolution verticale (nombre de points dans les Y),
- Son nombre de couleurs affichables en même temps.

Les modes graphiques sur VGA sont les suivants :

```
+-------------------------------------------------------------------+
|    n° | Mode               | Décodage :)                          |
|-----+-------------+-----------------------------------------------|
|      0 | 80x25x16        | Repasse dans le mode texte              |
|      7 | 320x200x16    | 320 pixels en X, 200 pixels en Y, 16 couleurs    |
|      8 | 640x200x16    | 640 pixels en X, 200 pixels en Y, 16 couleurs    |
|      9 | 640x350x16    | 640 pixels en X, 350 pixels en Y, 16 couleurs    |
|    10 | 640x480x4      | 640 pixels en X, 480 pixels en Y, 4 couleurs      |
|    11 | 640x480x2      | 640 pixels en X, 480 pixels en Y, monochrome      |
|    12 | 640x480x16    | 640 pixels en X, 480 pixels en Y, 16 couleurs    |
|    13 | 320x200x256 | 320 pixels en X, 200 pixels en Y, 256 couleurs |
+-------------------------------------------------------------------+
```

SCREEN

Le n° du mode est à transmettre à l'instruction SCREEN.

```
ex: SCREEN 13
```

Passera en mode 320x200x256

Quand je dis "passera" cela veut dire que vous verrez à l'éxecution de l'instruction SCREEN une saute puis un effacement de l'écran, cela revient en fait à une commutation de mode.

PSET

Une fois le mode graphique initialisé, vous avez droit à un petit paquet d'instructions dédiées au graphisme. Voyons tout d'abord l'instruction fondamentale : le point.

voici comment définir la couleur d'un point à l'écran : PSET (x, y), couleur

```
ex:PSET (20, 10), 13
```

Affichera un point rose en position 20, 10

Méfiez vous des coordonnées de PSET, elles commencent à zéro. Ainsi, en mode 13, 320x200, seules les coordonnées 0 à 319, 0 à 199 sont autorisées !

Notez que toute coordonnée en dessous du zéro ou au dessus des limites n'aboutit heureusement pas sur une erreur, qbasic détectant alors que le point est hors limites, il se passe donc simplement de l'afficher.

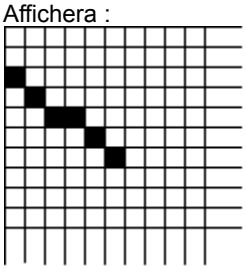
LINE

Pour dessiner des lignes, il y a l'instruction LINE. Voici quelle est sa syntaxe :

LINE (DebutX, DebutY)-(FinX, FinY), couleur

```
Exemple : LINE (0, 2)-(5, 6), couleur
```

Affichera :



Vous pouvez omettre les coordonnées de départ ou de fin. Si c'est le cas, Qbasic les remplacera par les coordonnées courantes.

Qu'est que les 'coordonnées courantes' ?

En fait, à chaque fois que vous tracez quelque chose à l'écran, qbasic déplace un curseur (purement virtuel) à l'endroit à dessiner, et ceci pour chaque point de la figure.

A la suite par exemple d'un PSET (5, 2), les nouvelles coordonnées courantes seront 5,2; tout simplement ! A la suite par exemple d'un LINE (10, 10)-(20,12) les nouvelles coordonnées courantes seront 20,12 car il s'agit du dernier point de la ligne !

```
Donc LINE (0, 0)-(20, 20), 1      suivie de
          LINE -(30, 30), 1
                     Il ne s'agit non pas d'un "moins" mais du tiret de
                        séparation des coordonées !!!

revient au même que :
          LINE (0, 0)-(30,30), 1
```

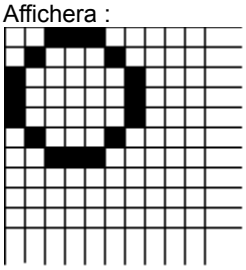
CIRCLE

Grâce à CIRCLE, il est possible de tracer Cercles, arcs de cercles et même camemberts pour les graphiques !

Syntaxe : CIRCLE (CentreX, CentreY), Rayon_en_pixels, couleur

```
ex:CIRCLE (3, 3), 3, couleur
```

Affichera :



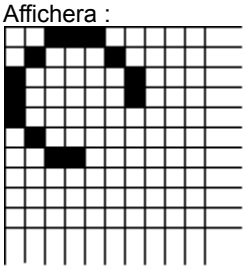
CIRCLE peut aussi afficher des arcs de cercles.

Syntaxe : CIRCLE (CentreX, CentreY), Rayon, couleur, AngleDebut, AngleFin

AngleDebut et AngleFin sont exprimés en radians, ce qui ne facilite bien sûr pas les choses. Voici pour faciliter les conversions une tit' fonction :

```
FUNCTION Radian! (degre%)
    Radian! = (3.141592 / 180) * degre%
END FUNCTION

ex:CIRCLE (3, 3), 3, couleur, Radian!(0), Radian!(270)
```

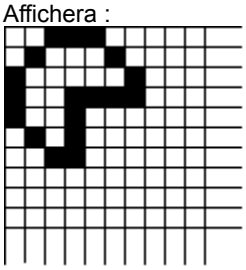
Affichera :



Vous pouvez aussi faire des camemberts facilement en rendant les angles de l'arc de cerle négatifs (ça n'a pas de logique en soi, la négativité est juste une indication à qbasic qui évite de rajouter d'autres paramètres à une instruction déjà bien longue !).

```
Ex:CIRCLE (3, 3), 3, couleur, -Radian!(0), -Radian!(270)
```

Affichera :



Dernière chose sur circle : Tout comme LINE ou PSET, tous ses paramètres sont facultatifs.

```
ex:CIRCLE (4,4), 3, , , Radian!(90)
  CIRCLE , 10, , , Radian!(180)
```

sont farpaitement corrects !

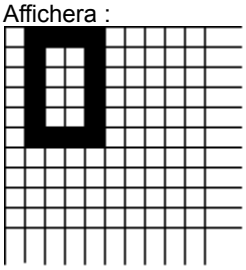
Rectangles

La primitive du rectangle s'obtient grâce à l'instruction LINE qui comme vous vous en doutiez à encore d'autres paramètres :

Syntaxe pour obtenir un rectangle : LINE (HautGaucheX, HautGaucheY)-(BasDroiteX, BasDroiteY), couleur, B

```
ex: LINE (1, 0)-(4,5), couleur, B
```

Affichera :

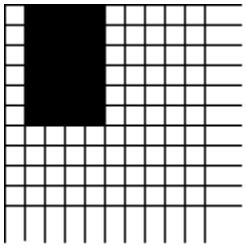


Donc dessiner un rectangle ne se distingue d'une ligne que par le petit paramètre "B" (B comme Box) de fin d'instruction !

En transformant ce B en BF, vous obtiendrez un rectangle plein (BF=Box Full).

```
ex: LINE (1, 0)-(4,7), couleur, BF
```

Affichera :



POINT

Vous savez afficher un pixel, voici maintenant comment lire la couleur de ce même pixel :

Syntaxe : couleur = POINT (x, y)

Exemple : Dans le graphique précédent,
POINT(0,0) renverra 0 et
POINT(2,2) renverra la couleur du rectangle.

Couleurs

Dans les modes graphiques 16 couleurs, vous retrouverez les attributs que vous connaissez bien (ceux du mode texte). Si dans une instruction graphique, vous ommetez le paramètre COULEUR, alors le qbasic prendra la couleur par défaut, celle attribuée via l'instruction COLOR.

Exemple :

```
COLOR 4
PSET (10, 10)
```

Affichera un pixel en (10,10) en rouge

PAINT

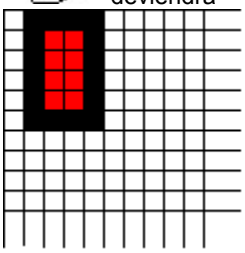
Cette instruction remplit une zone de l'écran avec une couleur.

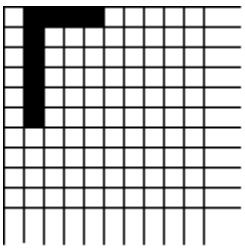
Syntaxe : PAINT (x, y), couleur_de_peinture, couleur_de_bordure

La couleur de bordure est celle où qbasic arrêtera le remplissage. Lorsque que PAINT est effectué dans une zone fermée, seule l'intérieur de la zone est remplit, en revanche, dans une zone ouverte, c'est tout l'écran qui risque d'être peint !!!

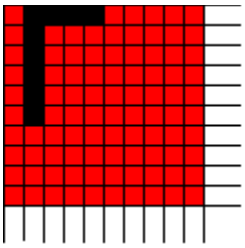
Après un PAINT (3, 2), 4, <couleur_de_bordure>

deviendra





deviendra



DRAW

Maintenant que vous connaissez toutes les primitives, voyons un tout autre type d'instruction graphique, qui vous inspirera peut-être, pour ceux qui connaissent (dans ce cas : désolé), le langage LOGO.

Donc tout part à partir du concept de coordonnées courantes (nous dirons curseur, mais pitié, ne faites pas de confusion avec le curseur du mode texte !).

En fait on donne des instructions de déplacement au curseur, comme "va à gauche", "va à droite", etc...

Voici les ordres :

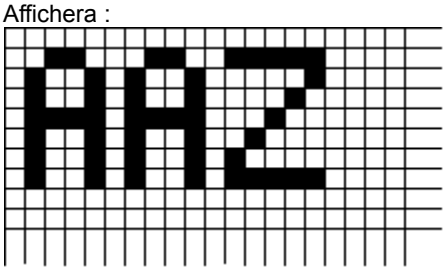
Ordre	Description	Paramètres
U	[Up] Trace vers le haut	<déplacement>
D	[Down] Trace vers le bas	<déplacement>
L	[Left] Trace vers la gauche	<déplacement>
R	[Right] Trace vers la droite	<déplacement>
E	Trace en haut à droite	<déplacement>
H	Trace en haut à gauche	<déplacement>
F	Trace en bas à droite	<déplacement>
G	Trace en bas à gauche	<déplacement>
N	Le cruseur revient à la position	aucun
B	Le curseur est déplacé sans tracer	aucun
M	Positionnement	<x,y>

Ces ordres contiennent de 0 à 2 paramètres et imposent un mouvement au curseur. La commande de positionement, M, permet de déplacer le curseur vers un point donné. Les commandes B et N sont des préfixes aux autres commandes, et indique respectivement qu'il faut se déplacer sans tracer ou qu'il faut revenir à la position initiale après le prochain tracé.

Tous les ordres se mettent les uns à la suite des autres dans une chaine passée à l'instruction ... DRAW !

```
Exemple : DRAW "BM1,7U5ERFD2NL2D3    BM6,7U5ERFD2NL2D3    BM11,1R4DG4DR4"
```

Affichera :



Cette instruction est extrêmement utile pour les fontes, les icônes ou les petits croquis.

Squirrel [Sep 95]

II. Jouer de la musique

Pour faire du son, nous avons déjà vu BEEP et SOUND, toutes deux peu aptes à faire de la belle musique melodieuse (à moins d'avoir une partition entière codée en hertz mais bon...).

Donc, voici ...

PLAY

PLAY fonctionne comme DRAW : Il y a quelques commandes, avec parfois un ou deux paramètres, qui sont mises les unes à la suite des autres dans une chaine.

Voici les commandes :

Ordre	Description
A	LA (Notes interntationales)
B	SI
C	DO
D	RE
E	MI
F	FA
G	SOL
+ ou #	La note indiquée est jouée un demi-ton plus haut
-	La note indiquée est jouée un demi-ton plus bas
>	Passe un octave plus haut
<	Passe un octave plus bas
O	Change l'octave actuel (suivi du n° de l'octave)
L	Fixe la durée des notes (de 1 à 64)
T	Fixe le nombre de noire par minute (de 32 à 255)
P	Fait une pause d'une durée entre 0 et 64
N	Joue n'importe quelle note de 1 à 84

Exemple : PLAY "CDEFGAB>C" jouera "DO RE MI FA SOL LA SI DO"

Si les commandes de notes (A à G), celles de dièse (#) et de bémol (-) sont claires (et même évidentes si vous aves fait un peu de solfège dans un pays anglosaxon), je m'en vais détailler les autres.

Les commandes > et < augmentent ou baissent d'une octave la note qui suit. Si par exemple l'octave en cours est 2, ">" passera à l'octave 3.

L'octave en cours est définissable via la commande O. "O4" passera à l'octave 4.

Maintenant, voyons un peu la durée des notes. Comme vous le savez (et si vous ne le savez pas, vous le saurez !), on attribue aux notes une durée. Voici comment les durées que vous connaissez sont convertibles pour fonctionner avec les commandes L et P :

1	ronde
2	blanche
4	noire
8	croche
16	double croche
24	triple croche
32	quadruple croche
etc... (jusqu'à 64)	

Exemple : "L4 CCCC" jouera 4 noires DO

La commande P elle aussi utilise ces durées.

```
Exemple : "P1" fera une pause silencieuse d'une ronde.
```

La commande T définit le nombre de noires que Qbasic doit cadencer en une minute, elle change en fait le tempo de la musique.

Enfin, la commande N joue n'importe quelle note de n'importe quel octave, entre 1 et 84. Etant donné qu'il y a 7 notes par octave, vous pouvez donc accéder à (84/7) 12 Octaves !

```
Exemple    :    "N84" jouera un SOL d'octave 12, comparable au bruit qu'emet
votre vieux moniteur VGA !
```

Ne vous réjouissez pas trop vite : Ce n'est pas fini !!

Il reste encore quelques commandes indispensables mais qui sont d'un tout autre ordre, elles servent à définir la façon dont les notes sont jouées.

```
+-----------------------------------------------------------+
| Ordre | Description
|
|-------+---------------------------------------------------|
|        MF | Joue les notes au 7/8 de leur durée                          |
|        ML | Joue les notes à leur pleine durée                            |
|        MS | Joue les notes au 3/4 de leur durée                          |
|        MF | Joue les notes en premier plan
|
|        MB | Joue les notes en arrière plan
|
+-----------------------------------------------------------+
```

Même chose, les commandes MF, ML, MS sont plutôt claires et influent sur le temps de pause entre chaque note. Par contre MF et MB elles, vous auront interloqué :

Le Qbasic dispose d'un petit buffer (16 notes, je crois me souvenir) qui permet de jouer les notes en arrière plan; il s'agit en fait, simplement, d'une musique de fond. Grâce à la commande MB, votre programme continuera à s'éxecuter sur un fond musical.

```
Exemple :
PLAY "MB L1 CDEFGAB"
PRINT "Vous verrez ce texte tout en entendant la musique"
```

Squirrel [Sep 95]

III. Votre programme

J'ai mis dans les annexes une démonstration récapitulant quelques une des fonctions vues dans cet article.

Maintenant, à vous de savoir utiliser tout ce qui a été vu et de le mettre à votre profit pour vous faire une belle interface graphique.

Comme cette fois ci je ne vous ai pas surchargé de boulot, pourquoi ne pas m'écrire une petite (toute petite hin, vous foulez surtout pas !) lettre pour me donner quelques conseils ? Ou même mieux, me poser quelques problèmes ou me montrer vos programmes !!

Errata

Squirrel [Sep 95]

IV. Errata

Quelques erreurs se sont glissées dans le précédent article (vous voyez à force de travailler sur les coups de minuit !!) :

J'ai mis qu'une syntaxe comme celle ci était autorisée :

```
TYPE machin
    PRENOM (1 TO 3) AS STRING*20
END TYPE
```

Et le pire, c'est que je le croyais !!! L'erreur, ici, c'est de mettre un tableau (1 TO 3) dans un TYPE, chose malheureusement formellement interdite en basic.

J'ai fait aussi une erreur d'inatention, j'ai mis :

"CHR$ est l'inverse de VAL"

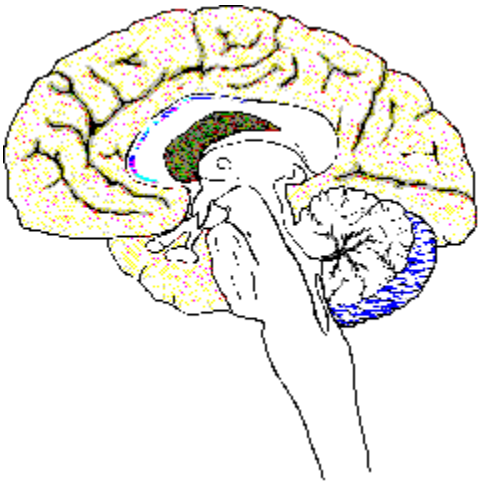
Vous aurez corrigé de vous même, CHR$ est l'inverse de ASC et non de VAL. VAL sert à transformer un nombre sous forme de STRING, en nombre sous forme numérique comme INTEGER.

Exemples :

```
chaine$ = "14.5"
s! = VAL (chaine$)
==> s! sera donc égale à 14.5

s$ = ASC (65)
==> s$ sera donc égale à "A".
```

La mémoire 2

Squirrel & Warrant [Aou 95]



La mémoire, second volet

Voici la seconde partie promise. Attention, cette fois ci, place aux programmeurs. Pour les débutants interessés par le premier volet et qui souhaitent une suite au premier article, écrivez moi et je vous concocterais quelque chose !

A cette deuxième partie, une autre suivra encore, parlant de la mémoire virtuelle, du tas, et peut être de disques virtuels.

Vous trouverez en annexe les programmes Pascal 7 suivants :

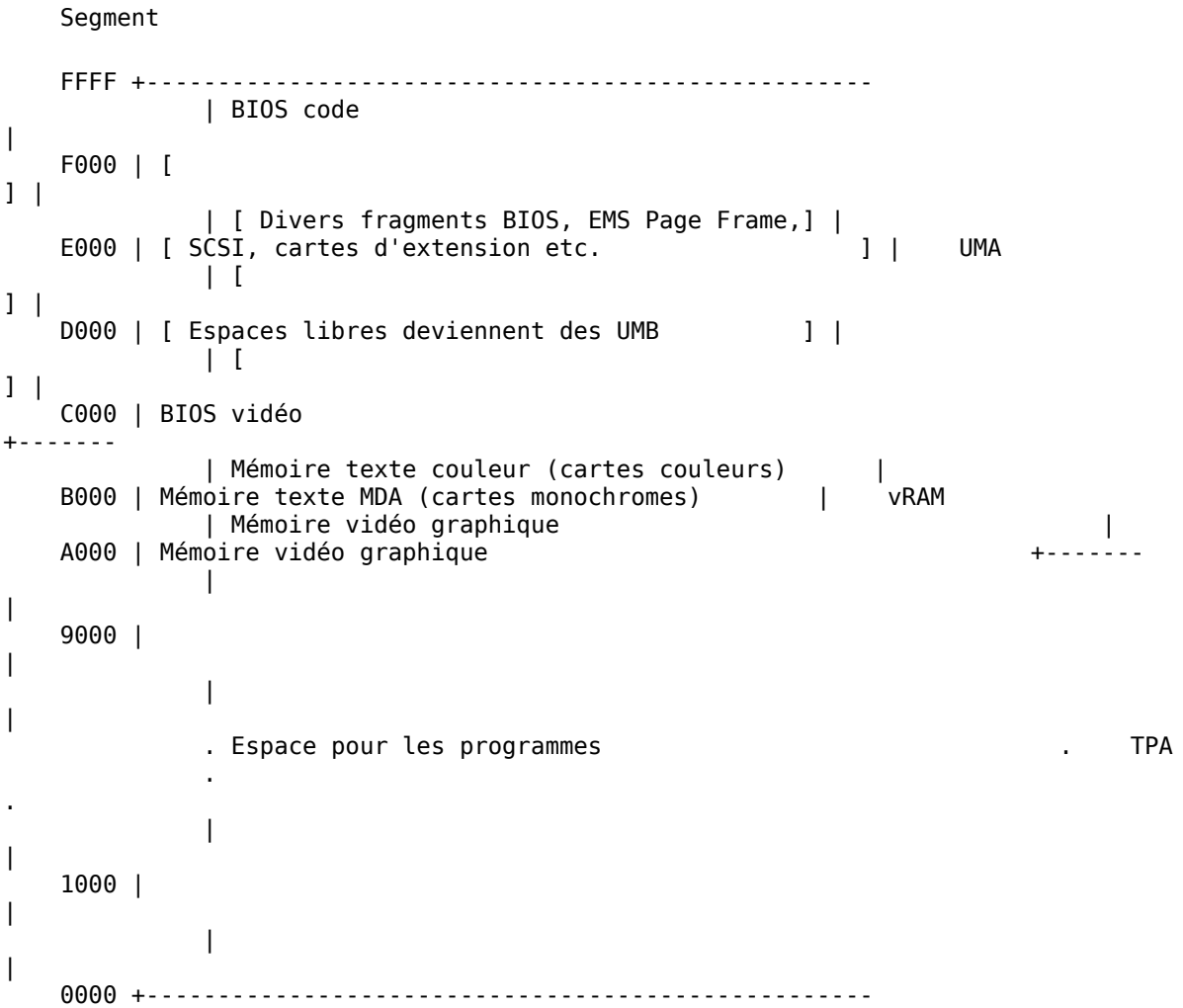
- Une unité comprenant une gestion de :
  - la XMS;
  - la HMA;
  - l'EMS;
  - des mcbs et de la mémoire externe dos;
  - des UMB;

- Des exemples de fonctionnement de l'unité.

Organisation de la mémoire

Squirrel [Aou 95]

1 a. Organisation de la mémoire conventionelle



Certaines adresses fixes sont très souvents utilisées en programmation car on y trouve des éléments vitaux du système.

C'est le cas par exemple du segment 0000 qui débute par 1024 octets, soit 256 pointeurs (oui je sais que vous savez qu'un pointeur fait 4 octets !) pointant sur les vecteurs d'interruption.

Au segment 0040h, il y a d'innombrables informations très très très pratiques pour ne pas dire indispensables sur le timer, l'affichage, les périphériques, le clavier, etc.

Tout savoir sur le segment 0040h !

Aux segments A000h, B000h, et B800h on trouve respectivement les adresses vidéos : graphique, texte noir & blanc, texte couleur.

Pour afficher à l'écran, écrire directement dans ces segments est infiniment plus rapide que de passer par des interruptions !

Si vous voulez en savoir plus sur les segments vidéos et la programmation graphique en général, jettez vous sur les géniaux articles de Rixed, Warrant et Mogar !

Enfin il y a quelques autres segments fixes utilisés de temps en temps tels que C000, F000, ou FFFF car ils contiennent à certaines adresses des choses précises, sur le BIOS notament.

Quand vous lancez un programme à partir du Dos, il va se placer dans la TPA -les 640 premiers ko- (TPA = Transient Program Area) et occupera la place dont il a besoin.

Pour connaitre constament la mémoire libre, le DOS maintient à jour une table des programmes qui indique leur taille et leur position dans la dite mémoire, cette table se présente sous la forme d'une liste chainée. Chacun des enregistrements de la liste (appelés MCB) indique non seulement la taille du programme mais aussi le programme suivant. Ainsi, il est possible au DOS d'allouer, désallouer, modifier des programmes ou leur taille respective.

Chaque programme est donc immédiatement précédé d'un des ces MCBs. Mais ce n'est pas tout, pour chaque programme le DOS crée aussi une structure appelée PSP (Program Segment Prefix) contenant beaucoup d'informations sur le programme, située juste à la suite du MCB.

Donc pour l'ensemble des programmes, que ce soit des .EXE ou des .COM, ces deux structures contiennent l'essentiel des informations qui le caractérise en mémoire.

Dans ce cas, quelle différence entre les .COM et les .EXE ?

Les programmes .COM sont des programmes dits binaires, ils ne contiennent aucune structure particulière pour leur chargement en mémoire, à l'inverse des .EXE.

Les .COM sont donc limités fatalement à la taille d'un segment (64ko), et leur code, leurs données, et leur pile doivent y cohabiter.

Les .EXE, eux n'ont pas cette contrainte : ils peuvent s'étendre sur plusieurs segments, généralement un pour les données, un ou plusieurs pour le code, et un pour la pile.

Mémoriquement parlant (arfarf), DOS attribue au démarrage des .COM l'ensemble de la mémoire encore libre, tandis qu'il alloue juste ce qu'il faut avec les .EXE.

Description du PSP

Squirrel [Aou 95]

Program Segment Prefix (PSP)		
00h	1 word	Contient un INT 20h qui quitte le programme
02h	1 word	Adresse de segment de la fin de la mémoire occupée par le programme
04h	1 byte	réservé
05h	1 word	appel de l'interruption 21h
0Ah	1 ptr	Copie du vecteur d'interruption 22h
0Eh	1 ptr	Copie du vecteur d'interruption 23h
12h	1 ptr	Copie du vecteur d'interruption 24h
16h	22 byte	Réservé
2Ch	1 word	Adresse de segment du bloc d'environnement
2Eh	46 byte	Réservé
5Ch	16 byte	FCB (File Control Block) #1
6Ch	20 byte	FCB #2
80h	1 byte	Taille de la ligne de commande
81h	127 byte	Ligne de commande

Description du bloc d'environnement

Squirrel [Aou 95]

Le bloc d'environnement contient l'ensemble des variables d'environnement affectables grâce à la commande DOS SET. En effet, les variables ne sont pas stockées en un seul endroit de la mémoire mais reportées pour chaque programme. A chaque éxecution d'un programme, le DOS recopie le bloc d'environnement du programme père dans celui du programme fils, et une fois le programme fils fini, son bloc d'environnement est tout simplement détruit, et non recopié dans le programme père.

Ainsi, il est impossible par voie normale de modifier les variables du programme père.

A partir du DOS 4.0, On trouve une nouvelle information dans le bloc d'environnement : Le nom complet du programme lié à ce bloc, avec chemin d'accès et extension.

Voici la structure du bloc d'environnement :

```
VARIABLE=Valeur + #0
... etc ...
```

La fin de la liste des variables d'environnement est marquée par un nouveau caractère #0 (soit deux caractères #0 en tout). Juste après ce second caractère #0, on trouve dans le DOS 4 le nom complet du programme associé, terminé également par #0.

Description du segment 40h

Squirrel [Aou 95]

Ah, mignon petit segment 40h.. Je m'en vais vous donner les plus intéressantes adresses du segment :

```
    0040:0010h W    contient un mot très très utile :
                                bit      1            : coprocesseur mathématique
installé
                                bits    6 et 7 : nombre de lecteurs de disquettes
                                                                    installés cela permet
par exemple
                                                                    d'éviter de rajouter le
lecteur B:
                                                                    dans une liste de
fichiers s'il n'y
                                                                    a qu'un seul lecteur !

    0040:0017h W    voici un renseignement complet de l'état des touches
                                du clavier :
                                bit      0            : Touche Shift droite
enfoncée
                                bit      1            : Touche Shift gauche
enfoncée
                                bit      2            : Touche Ctrl
enfoncée
                                bit      3            : Touche Alt
enfoncée
                                bit      4            : Led        Scroll Lock
activée
                                bit      5            : Led        Num
activée
                                bit      6            : Led        Caps
activée
                                bit      7            : Mode      Insert
activé
                                bit      8            : Touche Ctrl droite
enfoncée
                                bit      9            : Touche Alt    droite
enfoncée
                                bit    10            : Touche SysReq
enfoncée
                                bit    11            : Mode      Pause
activé
                                bit    12            : Touche Break
enfoncée
                                bit    13            : Touche Num Lock
enfoncée
                                bit    14            : Touche Caps Lock
enfoncée
                                bit    15            : Touche Insert
enfoncée

    0040:001Ah W    Indique l'offset du premier caractère dans le buffer
    0040:001Ch W    Idem mais le dernier caractère
                                Quand [1Ah] = [1Ch] alors buffer vide
    0040:001Eh        16 Words ici même qui constituent le buffer clavier.
                                Très pratique pour lire des touches sans les enlever
                                ou insérer des touches à l'insu d'un programme.

    0040:0049h B    Contient le mode graphique courant, valeur passée à
                                la fonction 00h de l'int 10h.

    0040:004Ah W    Nombre de colonnes à l'écran en mode texte
```

```
0040:004Ch W    Taille de la page d'écran en mode texte

0040:0063h W    Adresse du controleur graphique
                           Contient 3B4h si la carte est monochrome ou 3D4h si
                           elle est couleur.

0040:006Ch D    Un double mot 32 bits contenant le compteur horaire
                           en tops depuis minuit. Il y a environ 1092 tops par
                           minute, 18.2 par seconde.

0040:0084h B    Indique le nombre de lignes à l'écran en mode texte.
```

Voilà... Il y en a beaucoup d'autres mais si vous avez déjà fait le tour de celles là, je crois qu'il est grand temps pour vous de :

1. Vous acheter une bonne référence (genre "Pcinterdit" :-<) comme le livre d'or Sybex, la bible pc 3 ou 4 (la 5 beurk...) disponibles à la fnac du coin !

2. De venir écrire dans le Reporter !

Squirrel [Aou 95]

1 b. Allocation de vos propres zones

Le DOS, outre de proposer des fonctions pour démarrer un programme (et donc allouer la mémoire dont il a besoin), quitter un programme (et désallouer la mémoire qui lui avait été confiée), propose aussi des fonctions de gestion de la mémoire qui peuvent être très utiles au programme.

- Prenons le cas d'un programme de type .COM qui souhaite lancer un autre programme. Au démarrage du programme père, le DOS lui allouera toute la mémoire disponible, donc s'il ne libère pas ce dont il n'a pas besoin, l'éxecution du programme fils ne pourra aboutir, faute de mémoire suffisante.

- Prenons le cas maintenant d'un programme .COM ou .EXE qui a besoin d'un espace mémoire pour par exemple charger une image : il pourra faire appel au DOS pour obtenir une zone mémoire de la taille de l'image, avec laquelle il pourra ensuite travailler.

Dans les langages évolués, ce genre de problème n'arrive que rarement : D'abord ils ne compilent pas de programmes .COM, et ensuite les programmes .EXE constitués disposent de ce qui s'appele un TAS (ou heap, en anglais), sur lequel je reviendrais un peu plus loin dans ce chapitre.

Par contre, en assembleur, des cas comme ça ne manquent pas. Ainsi le DOS propose quelques fonctions très rudimentaires pour se charger de créer, supprimer ou modifier la taille de zones mémoire:

```
    +-----------------------------------------------------------------+
    | Réserver une zone mémoire
|
    |-----------------------------------------------------------------|
    | Nous utiliserons pour cela le service 48h du DOS :                          |
    | Entrée : MOV    AH, 48h
|
    |                    MOV    BX, <Nombre de paragraphes à réserver>
|
    | Sortie : JC      Erreur
|
    |                        AX contient l'adresse de segment de la zone mémoire    |
    |                        qui vient d'être allouée.
|
    |
|
    |                    Erreur:
|
    |                        BX contient le nombre de paragraphes encore dispos
|
    |                        AX le code d'erreur : 7 = MCB détruit
|
    |                                                                    8 = Mémoire
insuffisante              |
    |
|
    | Un paragraphe correspond à 16 octets.
|
    +-----------------------------------------------------------------+

    +-----------------------------------------------------------------+
    | Libérer une zone mémoire
|
    |-----------------------------------------------------------------|
    | Il s'agit là du service 49h, qui fait l'opération inverse du        |
    | précédent.
|
    |
|
    | Entrée : MOV    AH, 49h
|
    |                    MOV    ES, <Adresse de segment de la zone à libérer>
|
    | Sortie : JC      Erreur
|
    |                      Ici, tout va bien
|
    |
|
    |                    Erreur:
|
    |                        AX, code d'erreur : 7 = MCB détruit
|
    |                                                          9 = Adresse
invalide                         |
    |
|
    +-----------------------------------------------------------------+
```

```
    +-------------------------------------------------------------------+
    | Modifier la taille d'une zone mémoire
|
    |-------------------------------------------------------------------|
    | Très utile notament pour réduire la taille d'un programme .COM    |
    | en mémoire au strict minimum.
|
    |
|
    | Entrée : MOV    AH, 4Ah
|
    |                    MOV    BX, <Nouvelle taille en paragraphes>
|
    |                    MOV    ES, <Adresse de segment de la zone à libérer>
|
    | Sortie : JC      Erreur
|
    |                      Ici, tout va bien
|
    |
|
    |                    Erreur:
|
    |                        BX contient le nombre de paragraphes encore dispos
|
    |                        AX le code d'erreur : 7 = MCB détruit
|
    |                                                                  8 = Mémoire
insuffisante              |
    |                                                                  9 = Adresse
invalide                     |
    |
|
    +-------------------------------------------------------------------+
```

Vous pouvez vous poser une question bien justifiable : comment le DOS, lors d'une allocation, choisit-il le bon bloc à allouer ?

A partir de la **version 3**, le DOS propose en fait le choix entre plusieurs méthodes pour trouver LE bloc qui conviendra à la taille demandée par une allocation. Le choix d'une de ces méthodes est appelé Stratégie d'allocation.

Voici quelles sont les statégies offertes par le DOS :

- **0** = Prendre le premier bloc convenable à partir du début de la mémoire;
- **1** = Parcourir tous les blocs et choisir celui qui est le plus proche de la taille désirée;
- **2** = Prendre le premier bloc convenable à partir de la fin de la mémoire.

Voici comment communiquer sur les stratégies d'allocations du DOS :

```
    +-------------------------------------------------------------------+
    | Fixer la stratégie d'allocation du DOS (Dos 3+)
|
    |-------------------------------------------------------------------|
    | Entrée : MOV    AX, 5801h
|
    |                    MOV    BX, <Stratégie>
|
    | Sortie : JC      Erreur
|
    |                      Ici, tout va bien
|
    |
|
    |                    Erreur:
|
    |                        AX, code d'erreur : 1 = Code de fonction inconnu
|
    +-------------------------------------------------------------------+

    +-------------------------------------------------------------------+
    | Lire la stratégie d'allocation du DOS (Dos 3+)
|
    |-------------------------------------------------------------------|
    | Entrée : MOV    AX, 5800h
|
    | Sortie : JC      Erreur
|
    |                        AX contient la stratégie d'allocation
|
    |
|
    |                    Erreur:
|
    |                        AX le code d'erreur : 1 = Code de fonction inconnu
|
    +-------------------------------------------------------------------+
```

A partir de sa **version 5**, le DOS est apte à gérer viablement les UMBs. La Stratégie d'allocation s'affine alors permettant de privilégier une allocation dans les UMBs plutôt que dans la TPA.

Le bit 7 du code de stratégie d'allocation indique, lorsqu'il est sur 1 que la recherche de blocs dans les UMBs doit se faire en priorité. La TPA ne sera alors utilisée que s'il n'y a plus de blocs de taille suffisante dans les UMBs.

Ainsi, La stratégie BX=129 est sans doute la plus efficace. (priorité pour aux UMBs + recherche du bloc le plus proche).

Seulement, avant de profiter de cette nouvelle possibilité stratégique, il faudra auparavant vérifier que :

- Le DOS gère bel et bien des UMBs
- Que les UMBs sont biens inclus dans la gestion mémoire.

A cet effet, nous vient en aide le service 5802h du DOS :

```
    +-------------------------------------------------------------------+
    | Obtenir des informations sur les UMB (Dos 5+)
|
    |-------------------------------------------------------------------|
    | Entrée : MOV    AX, 5802h
|
    | Sortie : JC      Erreur
|
    |                       AL = 0    UMBs non utilisés
|
    |                             = 1    UMBs utilisés dans la gestion mémoire
|
    |
|
    |                     Erreur:
|
    |                         AX contient le code d'erreur :
|
    |                             = 7 Gestion de la mémoire détruite
|
    |                             = 1 UMBs non reconnus (non gérés)
|
    +-------------------------------------------------------------------+
```

Si cette fonction renvoie par exemple JNC et AL=1, alors les UMBs sont intégrés dans la gestion de la mémoire.

Si la fonction renvoie JNC et AL=0, alors il faudra d'abord intégrer les UMBs dans la gestion mémoire avant de pouvoir les utiliser. Pour demander à monsieur DOS de bien vouloir utiliser les UMBs, on passe par le service 5803h !

```
    +-------------------------------------------------------------------+
    | Déterminer l'utilisation des UMB (Dos 5+)
|
    |-------------------------------------------------------------------|
    | Entrée : MOV    AX, 5803h
|
    |                     MOV    BX, 1 pour inclure les UMBs dans la gestion
|
    |                                   0 pour ne pas les inclure
|
    | Sortie : JC      Erreur
|
    |                       Ici, tout va bien
|
    |
|
    |                     Erreur:
|
    |                         AX contient le code d'erreur :
|
    |                             = 7 Gestion de la mémoire détruite
|
    |                             = 1 UMBs non reconnus (non gérés)
|
    +-------------------------------------------------------------------+
```

Comment le DOS alloue t-il des UMBs ?

Voilà pour les UMBs.

Revenons sur le TAS. Le TAS est un concept présent notament dans les programmes compilés dans des langages de haut niveau tels que le pascal ou le c++.

Outre le code, la pile et le données, le compilateur crée un segment destiné au TAS, dans lequel le programmeur pour placer toutes sorte de données dites dynamiques.

Les fonctions DOS d'allocation et de désallocation MCB sont alors souvent superflues, car le programme dispose lui même, en interne de procédures gérant ce tas.

Bien sûr le tas a lui aussi une structure et des impératifs, ainsi que des procédures permettant sa gestion; mais ceci est un sujet suffisament vaste pour en faire un article complet, donc cela fera peut-être l'objet d'un Mémoire 3 ou 4 !! :-)

Comment DOS alloue t-il des UMBs ?

Squirrel [Aou 95]

Le DOS pour allouer des UMBs passe par deux fonctions offertes par le driver XMS (généralement HIMEM.SYS). Voici les dits services :

```
    +-------------------------------------------------------------------+
    | Allouer un UMB
|
    |-------------------------------------------------------------------|
    | Entrée : AH = 10h
|
    |                     DX = Taille du bloc mémoire à allouer (en paragraphes) |
    | Sortie : AX = 1     Ok, BX = Adresse de segment de l'UMB                    |
    |                     AX = 0    Erreur, BL = Codes d'erreurs XMS
|
    |                                                 DX = Taille du plus grand
UMB                   |
    +-------------------------------------------------------------------+
    +-------------------------------------------------------------------+
    | Libérer un UMB
|
    |-------------------------------------------------------------------|
    | Entrée : AH = 11h
|
    |                     DX = Adresse de segment de l'UMB
|
    | Sortie : AX = 1     Ok
|
    |                     AX = 0    Erreur, BL = Codes d'erreurs XMS
|
    +-------------------------------------------------------------------+
```

Ces services doivent être appelés par un CALL FAR sur l'adresse obtenue grâce au service 4310h du multiplexeur (int 2Fh). Pour plus d'informations sur ces fonctions, reportez vous à "gestion de la XMS" et "gestion de la HMA".

Codes d'erreurs XMS

Squirrel [Aou 95]

```
00 = Opération réussie
80 = Fonction spécifiée non connue
81 = Conflit avec un RamDisque
82 = Erreur sur l'addresse de ligne A20
8E = Erreur génerale du driver XMS
8F = Erreur système, impossible à éliminer
90 = HMA non trouvée
91 = HMA déjà occupée
92 = Seuil d'allocation trop bas
93 = HMA non allouée
94 = La ligne A20 est encore en fonction
A0 = Plus de mémoire étendue disponible
A1 = Tous les Handles XMS sont occupés
A2 = Accès à une zone [Handle] indéfinie
A3 = Erreur de l'handle source
A4 = Erreur de l'offset source
A5 = Erreur de l'handle cible
A6 = Erreur de l'offset cible
A7 = Déplacement - longueur incorrecte
A8 = Déplacement - Superposition interdite
A9 = Erreur de parité
AA = UMB non verrouillé
AB = UMB encore vérouillé
AC = UMB - Débordement de verrouillage
AD = UMB non verrouillable
B0 = UMB plus petit disponible
B1 = Plus d''UMB Disponible
B2 = Adresse de segment d'UMB incorrecte
```

Fonctionnement des MCBs

Warrant [Aou 95]

1 c. Fonctionnement des MCB

Ecrit par WARRANT :

Voilà donc ma contribution à l'article sur la mémoire, écrit en collaboration avec l'écureuil digital. Au fait, si vous ne trouvez pas l' "ASM par la pratique 4", c'est normal: plus le temps pour cette fois !

Disclaimer: les informations présentes dans cet article ne sont données que dans un but purement informatif. Tout dommage causé par la mauvaise utilisation de ces informations n'engage en aucun cas la responsabilité de l'auteur. (j'ai toujours adoré ces daubes de disclaimers ,pas vous ?)

Les MCB sont probablement ce qu'il y a de plus intéressant dans la gestion mémoire du DOS. Et pour cause: toute la gestion mémoire du DOS se fait avec les MCB !

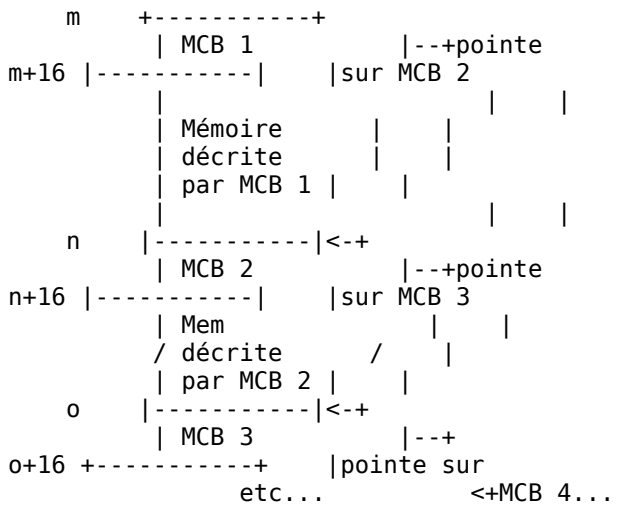
Quand vous manipulez de la mémoire avec les fonctions 48h, 49h et 4ah de l'int 21h, vous définissez / éliminez / ajustez des blocs de mémoire.

Ainsi,lors de la réservation de mémoire, vous obtenez une adresse mémoire, à partir de laquelle vous allez pouvoir procéder à toutes les manipulations possibles et imaginables (à condition de ne pas déborder de votre bloc!)

Une fois votre bloc réservé, le DOS place immédiatement avant celui ci une structure de 16 octets, chargée de le décrire. Les blocs ont eux une taille évidemment variable.

De plus, les MCB fonctionnent selon le principe d'une liste chainée: chaque MCB pointe sur le suivant. (cf Fig.666)

Fig.666: Principe du chaînage des MCB en mémoire



Pour pouvoir pointer sur le MCB suivant, il faut bien entendu connaitre la taille du bloc mémoire qui les séparent. Il faut aussi savoir si l'on ne se trouvait pas dans le dernier bloc mémoire réservé. Toutes ces infos sont stockées dans les MCB :

```
   Structure d'un MCB: 00= Identité (M ou Z, voir plus loin) [1 byte]
   ------------------    01= Adresse de segment PSP                        [1 word]
                                      03= Nombre de paragraphes réservés
[1 word]
                                      05= Non documenté
[11 bytes]
```

Le premier octet (identité) sert à savoir si le MCB est le dernier de la liste,ou si il pointe sur un suivant.

```
     'M' = il reste des blocs après celui-ci
     'Z' = dernier bloc réservé.
```

'M' et 'Z' seraient les initiales de 'Mark Zbikowsky', un dévelop- peur de Microchiotte. (Intermède culturel: dans les modules "protracker", les initiales "M.K." correspondent à "Mahoney" et "Kaktus", les concepteurs du premier soundtracker. Fin de l'intermède culturel.)

Le quatrième octet donne le nombre de paragraphes réservés. Pour avoir la taille en octets,il faut multiplier par 16. Pour savoir où se trouve le MCB suivant (si il y en a un),il suffit de se placer à la fin du MCB, et d'additionner cette valeur. On tombe alors sur le début du MCB suivant.

Bon, c'est bien beau tout ça, mais comment accéder à ces fameux MCB ? C'est trèès simple !! Ils se trouvent 16 octets avant le PSP. Comme ES pointe sur le PSP lors de l'initilisation d'un programme, il devient facile de récupérer l'adresse du MCB associé au programme.

```
(tout début du programme)-> DEC ES ;se place au début du MCB du prg
                                                    -> MOV MCB_PTR,ES
```

Ah oui, mais si on veut récupérer un MCB qui précède l'actuel dans la mémoire ? La liste chaînée ne fonctionne que dans un sens, non ? Ben oui. Nous allons donc en être réduit à utiliser une fonction non documentée du DOS. (arf,non-documentée,mais tellement utilisée qu'il n'y a pas de danger pour qu'elle disparaisse)

Cette fonction permets d'obtenir l'adresse du DIB (Dos Information Bloc). Le DIB contient toutes sortes d'infos à usage interne, et en particulier l'adresse du premier MCB !

Pour obtenir l'adresse mémoire où l'on trouvera le DIB, il faut utiliser la fonction 52h de l'interruption 21h. En retour, dans ES:BX , on a l'adresse du DIB. Hélas, l'adresse contenue dans ES:BX ne pointe pas sur le début du DIB ! Pour obtenir ce qui nous interesse (l'addy mémoire du premier MCB), il faut se placer
4 octets avant l'adresse retournée !
"Boah, il suffit de faire une bête soustraction!". Ca ne suffit pas. Si par exemple "Get_DIB" vous retourne l'adresse 3000:0003, et que vous soustrayez 4 à BX, vous allez obtenir 3000:FFFF (inutile de dire que vous n'êtes plus vraiment dans le DIB :))

Bon en fait, c'est tout (.....) <- complétez par un adjectif exprimant une facilité écoeurante, j'en ai marre de taper "simple" 15.000 fois par article.

Voici ce qu'il suffit de faire:

```
Mov ah,52h        ;Get DIB addy
Int 21h
Dec ES               ;on se retrouve 16 octets en arrière
Add BX,12          ;(16-4) = 12
```

Et hop, ES:BX pointent désormais sur l'adresse mémoire contenant le premier MCB. (Ah oui,attention hein! Au risque d'avoir l'air d'insister lourdement, ES:BX est un pointeur sur une adresse)

Avec ça, vous en savez suffisament pour explorer votre allocation mémoire, et savoir enfin ce que le DOS y planque dès que vous avez le ... dos tourné!

L'intêret de manipuler les MCB ? Heuuu... Franchement je ne vois pas...

A ma connaissance, les seules personnes manipulant directement les MCB sont les concepteurs de virus. En effet, la plupart des AntiVirus s'empressent de détourner l'int 21h du DOS,et de scruter tout particulièrement les appels aux fonctions mémoire (pour le disk, c'est plutôt l'int 13h).

Il devient donc intéressant de passer directement par les MCB pour se réserver de la mémoire, et d'éviter ainsi des interruptions "fliquées".

Certains lamers continuent cependant à coder des virus utilisants les fonctions du DOS pour gêrer la mémoire. (les bêbêtes de "naissance d'un virus" fonctionnent selon ce principe)

Remarquez que vous pouvez aussi vous prémunir des virus en écrivant un programme qui vous donne l'allocation mémoire, et que vous pouvez lancer AVANT et APRES avoir exécuté un programme "louche" : si après l'exécution du programme "louche" vous obtenez un [ou plusieurs] MCB supplémentaires, le programme en question est peut-être un virus. Attention à ne pas vous affoler trop vite, le programme en question mets peut-être simplement une chaîne d'environnement en mémoire, ou bien c'est tout simplement un résident. Enfin là, ça devient une affaire entre vous et votre débugger (ou votre v-scan !).

Les finesses des MCBs

Squirrel [Aou 95]

1 d. Les finesses des MCB

Suite à l'interrogation de Warrant :

La manipulation des MCBs peut se réveler indispensable ou pratique dans de nombreux cas : Modification d'environnements, antivirus, tsr, dumper, util du genre MEM, swapexec, lanceur d'éxecutable etc.

Les MCB sont des petites bêtes qui comme le veut la règle, elles aussi ont connues des améliorations (mais toujours ascendamment compatible), que je vais m'empresser de détailler :

Avec le dos 4, la structure des MCB s'étend :

```
+-----------------------------------------------+
| Memory Control Block dos 4.x                  |
|-----------------------------------------------|
| 00h | 1 byte | Identificateur (M ou Z)        |
| 01h | 1 word | Adresse du PSP associé         |
| 03h | 1 word | Taille de la zone en paragraphes |
| 05h | 3 byte | Inutilisé                      |
| 08h | 8 byte | Nom du programme associé       |
+-----------------------------------------------+
```

Vous avez remarqué l'apparition du champ 'nom de programme', il s'agit là du DOS qui copie les 8 premières lettres de l'éxecutable chargé en mémoire. Si le nom fait moins de huit lettres, il rajoute un caractère 0. Ce champ est très utile notament dans les TSR pour obtenir simplement le nom du programme tournant avant l'appel du TSR.

Avant d'utiliser ce nom, il faut vérifier la nature du MCB. Voici quels sont les différentes natures de MCB :

- **Programme** : La zone définie par le mcb est reservée à un programme.

- **Environnement** : Il s'agit des variables DOS (affectables grâces à la commande SET. C'est le cas par exemple de PATH, PROMPT ou les variables CONFIG ou COMSPEC) relatives à un programme. Les environnements de programmes sont générallement tous de petite taille.

- **Non identifiable** : Il s'agit probablement d'une zone allouée par un programme.

Le champ 'nom' n'est exploitable que si le mcb contient un programme. Pour le vérifier, voici une méthode :

Si l'adresse de segment du mcb + 1 (pointant donc sur le début du bloc) est équivalent à celle du psp (champ 01h, psp associé) on peut être convaincu de la présence du champ 08h (nom).

SI mcb.seg+1 = mcb^.psp_assoc ALORS [c'est un programme]

Pour identifier un environnement, c'est hautement plus compliqué. Il faut faire une liste de tous les programmes, et vérifier pour chaque MCB s'il correspond au champ 'Environnement associé' de chaque programme identifié ! D'autres méthodes telles que la vérification de la commande COMSPEC= devant chaque bloc font légion (même dans la Bible Pc) mais ne fonctionnent pas, notament si l'utilisateur a un menu DOS 6 multi-config ou a fait appel à SET dans CONFIG.SYS...

Avec le dos 5 et l'arrivé des UMB,

une chose change fondamentallement :

Il peut y avoir d'autres MCBs après l'identificateur "Z" ! Si tel est le cas, les MCBs porteront l'id "M" jusqu'à tomber à nouveau sur un "Z" indiquant alors la réelle fin des mcbs.

Mais avant de parcourir tous ces MCBs en attendant de passer 2 fois l'id "Z", mieux vaut vérifier au moins une chose : Que le mcb suivant le premier "Z" est bien valide.

Pour ceci, il faut vérifier que son id est soit "M" ou "Z".

Après le premier "Z", on trouve très souvent un MCB placé juste avant la mémoire vidéo (en A000-1:0000, B000-1:0000 ou B800-1:0000 selon que vous ayez désactivé telle ou telle partie de la mémoire vidéo). Ce MCB couvre tout le reste de la mémoire vidéo, et très souvent les données BIOS vidéo en C000:0000.

Ensuite, vous trouverez de nombreux MCB tels que celui ci recouvrant des zones systèmes BIOS. DOS les établis pour garder la cohésion entres les différentes zones vierges et ne pas empiéter sur ces données importantes.

En savoir encore plus sur les MCB (ahhhhhhhhhh....) !

MCBs relatifs aux drivers

Squirrel [Aou 95]

C'est incroyable : avec Microsoft, on n'en fini pas d'en finir tellement il y a de structures, de champs, de pointeurs sur pointeurs pointant sur des pointeurs...

Bon, je vais juste parler de ce qu'il convient d'appeler une seconde souche de MCB. Dos gère de façon permanente deux listes de MCB :

L'une réservée aux programmes DOS dont nous avons longuement parlé, L'autre réservée aux "Device Drivers".

Les Device Drivers (ou Drivers de périphérique) sont des petits programmes organisés d'une manière très particulière et qui sont aptes à gérer un périphérique. Dans le DOS bas niveau, les périphériques sont par exemple CON, NUL, CLOCK$, LPTx, COMx ou encore XMSXXXX0 (chargé de la XMS), EMSXXXX0, STACKER etc etc.

Ces petits programmes sont implémentés soit par le DOS, soit par des drivers externes (ANSI prend en charge le périphérique CON, HIMEM le XMSXXXX0 etc.)

Tout ces drivers sont reliés entre eux par des DDH ou Device Driver Header.

```
        +-------------------------------------------------------------+
        | Device Driver Header dos 2.x
|
        |-------------------------------------------------------------|
        | 00h | 1 ptr    | Pointe sur le prochain DDH
|
        | 04h | 1 word | attributs du driver
|
        | 06h | 1 word | Point d'entrée de la routine de stratégie           |
        | 08h | 1 word | Point d'entrée de la routine d'interruption        |
        | 0Ah | 8 byte | Nom du driver de périphérique
|
        +-------------------------------------------------------------+
```

Le nom du driver de périphérique n'est exploitable que si le bit 15 des attributs du driver est fixé (indiquant un driver de caractère).

Vous pouvez parcourir tous les DDH en utilisant le champ 00h; lorsque DDH.ofs = 0FFFFh alors il n'y a plus de DDH.

Pour obtenir l'adresse du premier DDH, il suffit encore une fois de passer par la fonction 52h de l'API DOS !

Le premier driver est situé à l'adresse ES:BX+22h retournée. Enfin ce n'est pas tout à fait aussi simple, car suivant les versions du Dos cette adresse change :

```
        +--------------------+        +-------------------------------------+
        | Ver. |      Adresse      |        | A toutes fins utiles, voici comment |
        |------+-------------|        | obtenir la version dos :
|
        | 2.x    | ES:BX + 17h |        |          mov ah, 30h    puis    int 21h
|
        | 3.0    | ES:BX + 28h |        | ==> ax contient alors la version        |
        | 3.1+ | ES:BX + 22h |        | ex: avec dos 3.1, ax contient 0301h |
        +--------------------+        +-------------------------------------+
```

Bon ok, mais quel rapport avec les MCBs ??

A partir du DOS 4.x il peut se cacher un MCB derrière un DDH ! L'adresse du MCB s'obtient en soustrayant 1 à l'adresse de segment du DDH, tout simplement ! Il suffit ensuite de vérifier que l'adresse de 'début du psp' pointe effectivement sur le DDH et on est assuré de la présence d'un MCB. Il est possible grâce à cela d'en savoir plus sur un driver de périphérique, et notament de savoir quel programme en est le propriétaire..

Ah, mais à quoi ça sert ?

Je ne vous cacherais pas que je vois dans cet deuxième souche de MCB une utilité très restreinte.. A la limite cela peut être utile aux programmes du genre SI, MSD ou MEM...

Je vous en ai parlé pour le souci d'être complet.. et on ne sait jamais cela servira peut-être à apaiser l'un d'entres vous, un jour, lorsque il sera épris d'une nérvosité extrême, arrachant une à une les touches de son beau clavier micro$oft natural keyboard (beurk !) parce qu'il n'arrive pas à implanter les structures bas niveau de son super driver type RAMDRIVE.SYS !

Attributs d'un driver dans le DDH

Squirrel [Aou 95]

Voici la structure complète, ca peut toujours servir.

```
    +-------------------------------------------------------------------+
    | Device Driver Header dos 2.x
|
    |-------------------------------------------------------------------|
    | 00h | 1 ptr    | Pointe sur le prochain DDH
|
    | 04h | 1 word | attributs du driver (détaillés plus loin)               |
    | 06h | 1 word | Point d'entrée de la routine de stratégie               |
    | 08h | 1 word | Point d'entrée de la routine d'interruption           |
    | 0Ah | 8 byte | Nom du driver de périphérique
|
    |-----+--------+-------------------------------------------------|
    | 12h | 1 word | CD-ROM Réservé = 0000h
|
    | 14h | 1 byte | CD-ROM Lettre du lecteur
|
    | 15h | 1 byte | CD-ROM Nombre d'unités
|
    | 16h | 4 byte | CD-ROM Id = 'MSCD'
|
    | 1Ah | 2 byte | CD-ROM Version au format ascii
|
    +-------------------------------------------------------------------+
```

Voici la signification des attributs du driver.

```
Bit    15 = 1    Driver de caractère :

                          14 = fonctions dos api IOCTL supportées
                          13 = sortie jusqu'à occupation supportée
                          12 = réservé
                          11 = OPEN/CLOSE/REMMEDIA supportés
                       10-7 = réservés
                             6 = fonctions génériques IOCTL supportées
                             5 = réservé
                             4 = Implique l'utilisation de l'int 29h
                             3 = Périphérique = CLOCK$
                             2 = Périphérique = NUL
                             1 = Sorties standarts
                             0 = Entrées standarts


           15 = 0    Driver de bloc :

                          14 = fonctions dos api IOCTL supportées
                          13 = Format non-IBM
                          12 = réservé
                          11 = OPEN/CLOSE/REMMEDIA supportés
                       10-7 = réservés
                             6 = fonctions génériques IOCTL supportées
                         5-2 = réservés
                             1 = adressage 32 bits secteurs supportée
                             0 = réservé
```

Les champs CD-ROM ne sont valables que si les 4 premiers octets d'identification concordent avec le texte 'MSCD'.

Squirrel [Aou 95]

2. Gestion de la HMA

Pourquoi a t-on fait la distinction des 64 premiers ko du second Méga et non des suivants ??

Car ils sont très spéciaux ces 64ko : il est possible d'y accéder directement, en mode réel, comme s'il s'agissait de n'importe quel autre morceau de mémoire !!

En fait, cette étonnante chose est le fruit de la méthode utilisée pour adresser la mémoire : Un segment multiplicateur par 16 allant de 0 à 0FFFFh (65535) et un offset, qui contient le déplacement par rapport au segment et allant lui aussi de 0 à
0FFFFh.

Si par exemple on choisit le segment FFFFh, il n'y aura que 16 octets (et donc offsets) correspondants au premier méga :

```
<taille de la mémoire> - <dernier segement> = 16
1024*1024                          - FFFFh*16                    = 16
1'048'576                          - 1'048'560                  = 16
```

Seulement si on s'amuse à adresser avec un offset supérieur ou égal à 16, on ne va plus accéder au premier méga, mais au second !

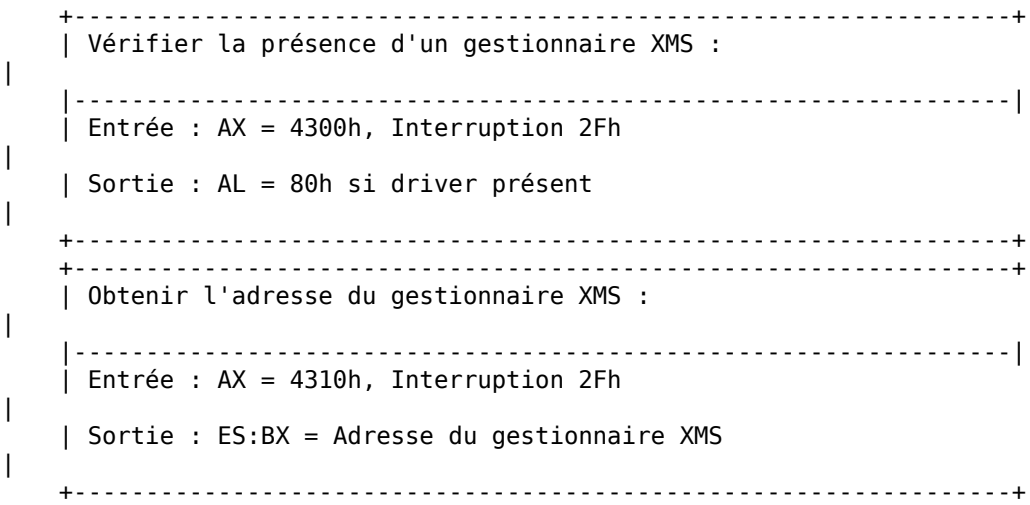
Il y a ainsi 65520 (64ko-16) octets du Second méga qui sont pleinement adressables !!

Mais voilà, pour pouvoir profiter de cette finesse, il faut vérifier qu'il y ait bien un gestionnaire XMS (C'est HIMEM.SYS qui se charge de la gestion de la HMA), et qu'il confirme que la HMA soit libre (A partir du DOS 5, le noyau du DOS peut occuper la HMA avec la commande DOS=HIGH de CONFIG.SYS).

Dans le cas où la HMA serait libre d'utilisation, il faut alors penser à débloquer la ligne A20 (20 est le numéro de la pin du processeur !) pour pouvoir y accéder directement, en mode réel. Cela se fait très simplement grâce au même driver XMS.

Voici un récapitulatif complet des fonctions relatives à la HMA pour vous permettre d'y accéder :

Le gestionnaire d'XMS/HMA HIMEM.SYS se 'branche' sur le multiplexeur pour permettre d'obtenir l'adresse d'appel de ses services.



A partir d'ici, les services sont appelés grâce à un CALL FAR sur l'adresse ES:BX renvoyée par le multiplexeur.

```
    +-------------------------------------------------------------------+
    | Obtenir le numéro de version XMS :
|
    |-------------------------------------------------------------------|
    | Entrée : AH = 00h
|
    | Sortie : AX = Numéro de version XMS
|
    |                    BX = Numéro de révision interne
|
    |                    DX = 0 si accès à la HMA impossible
|
    |                         = 1 si HMA disponible
|
    +-------------------------------------------------------------------+
    +-------------------------------------------------------------------+
    | Obtenir la HMA :
|
    |-------------------------------------------------------------------|
    | Entrée : AH = 01h
|
    |                    DX = Taille demandée (Mieux vaut mettre FFFFh, car
|
    |                              car si DX est plus petit que le paramètre
|
    |                              /HMAMIN de HIMEM.SYS, alors le gestionnaire vous
|
    |                              refusera de toutes façons l'obtention de la HMA.
|
    | Sortie : AX = 1 si fonction bien executée
|
    |                         = 0 si erreur, dans ce cas BL=Codes d'erreurs XMS
|
    +-------------------------------------------------------------------+
    +-------------------------------------------------------------------+
    | Libérer la HMA :
|
    |-------------------------------------------------------------------|
    | Entrée : AH = 02h
|
    | Sortie : AX = 1 si fonction bien executée
|
    |                         = 0 si erreur, dans ce cas BL = Codes d'erreurs XMS |
    +-------------------------------------------------------------------+
```

La ligne A20 est chargée d'autoriser l'accès à la HMA en mode réel. Lorsque elle est activée, déverrouillée ou libérée (qui veulent dire la même chose dans ce cas !), l'accès à la HMA est autorisé. Si elle est désactivée, verrouillée ou fermée l'accès est alors (et c'est logique !) impossible !

```
    +-----------------------------------------------------------------+
    | Activer l'adresse A20 :
|
    |-----------------------------------------------------------------|
    | Entrée : AH = 03h
|
    | Sortie : AX = 1 si fonction bien executée
|
    |                          = 0 si erreur, dans ce cas BL = Codes d'erreurs XMS |
    +-----------------------------------------------------------------+
    +-----------------------------------------------------------------+
    | Désactiver la ligne A20 :
|
    |-----------------------------------------------------------------|
    | Entrée : AH = 04h
|
    | Sortie : AX = 1 si fonction bien executée
|
    |                          = 0 si erreur, dans ce cas BL = Codes d'erreurs XMS |
    +-----------------------------------------------------------------+
```

Ces fonctions, dites globales, ont un effet immédiat sur la ligne A20. Elles sont doublées par les fonctions dites locales 05h et 06h qui quant à elles facilitent les activations/désactivations au sein de sous-programmes. L'avantage de ces deux autres fonctions locales est que si la ligne A20 est verrouillée 3 fois, elle devra être dévérouillée 3 fois pour de nouveau être active, chose tout particulièrement utile dans les sous-programmes.

Dernier service,

```
    +-----------------------------------------------------------------+
    | Obtenir l'état de la ligne A20 :
|
    |-----------------------------------------------------------------|
    | Entrée : AH = 07h
|
    | Sortie : AX = 1 si A20 libre
|
    |                          = 0 si A20 fermée
|
    +-----------------------------------------------------------------+
```

3. Gestion de la XMS

Ultiman vous a fait un superbe article sur le sujet dans le reporter n°4, alors inutile pour moi d'y revenir !

Squirrel [Aou 95]

4. Gestion de l'EMS

L'EMS a beau être une ancienne norme, elle trouve encore souvent son application dans les logiciels d'aujourd'hui. La transition avec la XMS ne s'est pas totalement faite pour la raison que l'EMS a des avantages de taille dans sa gestion.

En effet, quand avec l'XMS il faut forcément passer par un buffer pour lire et écrire des informations, il est possible avec l'EMS d'y écrire directement.

Pour cela, tout passe par le EMS PAGE FRAME, ou Cadre de Page EMS. Il s'agit d'une fenêtre de 64ko, généralement placée dans l'UMA; et elle porte bien son nom, cette fenêtre, puisque elle donne un accès immédiat sur n'importe quel partie de la mémoire étendue. En fait, elle serait comparable à un ascenceur qui permet d'accéder pleinement et en même temps à un seul étage du gigantesque building qu'est la mémoire étendue.

L'unité de base pour l'allocation de mémoire EMS est la page. Une page fait exactement 16ko, ce qui permet d'en loger 4 dans l'EMS PAGE FRAME.

Lorsque le programmeur à besoin de mémoire EMS, il alloue une zone mémoire pouvant contenir de une à plusieurs de ces pages. Ensuite, pour y accéder, il substitue cette zone à l'EMS PAGE FRAME de telle façon qu'un accès direct à celle-ci soit possible. Bien sûr il y a moyen de faire correspondre par exemple la 2° page de notre zone mémoire EMS à la 1° page du EMS PAGE FRAME...

Tous ces services sont regroupés dans l'interruption 67h, destinée au gestionnaire EMS (EMM386, 386MAX, QEMM386, etc...).

Mais avant d'appeler naïvement cette interruption, mieux vaut s'assurer qu'il y a bien quelque chose derrière, sinon le plantage sera très probable. Malheureusement, le driver EMS ne nous facilite pas la tâche.

Heureusement, c'est un driver, il est donc précédé d'un DDH (Device driver header), il va donc être possible à l'aide de ce DDH d'identifier l'interruption 67h comme appartenant à un driver EMS.

Pour ceci, il faut savoir que le nom du driver EMS est 'EMMXXXX0'. Donc si le champ 'nom' (offset 10d) du DDH présent à l'adresse <segment du vecteur d'interruption 67h>:0 est égal à 'EMMXXXX0' alors on a la certitude de la présence et du fonctionnement du driver EMS (ou EMM, soit Expanded Memory Manager).

+-> Exemple de détection d'un driver EMS

```
|
| Function IsEmsDriver : boolean;
| TYPE EmmName = array[1..8] of char;
|            EmmPtr    = ^EmmName;
| VAR    reg : registers;
| BEGIN
|      reg.ah := $35;
|      reg.al := EmsProc;
|      MsDos(Reg);
|      IsEmsDriver := (EmmPtr(Ptr(Reg.ES,10))^ = 'EMMXXXX0');
| END;
```

Voici maintenant la liste des services EMS 3.0 (les caractéristiques des versions suivantes sont rarement utilisées), appelables grâce à l'interruption 67h :

| ◆ Déterminer l'état de l'EMM
| IN : AH=40h
| OUT: AH=Etat de l'EMM (voir Codes d'erreurs EMS)

| ◆ Obtenir l'adresse de segment de l'EMS PAGE FRAME
| IN : AH=41h
| OUT: AH=Code d'erreur (0=Ok)
| BX=Adresse de segment du PAGE FRAME

| ◆ Déterminer le nombre de pages EMS libres
| IN : AH=42h
| OUT: AH=Code d'erreur (0=Ok)
| BX=Nombre de pages libres
| DX=Nombre de pages totales libres ou occupées

| ◆ Allouer mémoire EMS
| IN : AH=43h
| BX=Nombre de pages (de 16ko) à allouer
| OUT: AH=Code d'erreur (0=Ok)
| DX=Handle pour l'accès à la mémoire allouée

| ◆ Permet de substituer une page logique à une des pages physiques
| de l'EMS page FRAME (Fixer le Mapping)
| IN : AH=44h
| AL=Numéro de la page physique CIBLE
| BX=Numéro de la page logique SOURCE
| DX=Handle source
| OUT: AH=Code d'erreur (0=Ok)

| ◆ Libérer de la mémoire EMS
| IN : AH=45h
| DX=Handle
| OUT: AH=Code d'erreur (0=Ok)

| ◆ Obtenir la version de l'EMM
| IN : AH=46h
| OUT: AH=Code d'erreur (0=Ok)
| AL=Numéro de version EMM (forme BCD)

| ◆ Sauvegarder le contexte mapping.
| L'ensemble des correspondances logiques/physique est sauvé.
| IN : AH=47h
| DX=Handle
| OUT: AH=Code d'erreur (0=Ok)

| ◆ Restaurer le contexte mapping.
| L'ensemble des correspondances logiques/physique est sauvé.
| IN : AH=48h
| DX=Handle
| OUT: AH=Code d'erreur (0=Ok)

| ◆ Déterminer le nombre de Handles attribués
| IN : AH=4Bh
| OUT: AH=Code d'erreur (0=Ok)
| BX=Nombre de handle attribués

| ◆ Déterminer le nombre de pages allouées pour un handle

```
|      IN : AH=4Ch
|                DX=Handle à détailler
|      OUT: AH=Code d'erreur (0=Ok)
|                BX=Nombre de pages logiques attribuées

| ◆ Déterminer le nombre de pages allouées pour chaque handle
|      IN : AH=4Ch
|                ES:DI=Pointeur sur le tableau de Handle/nbre de pages
|      OUT: AH=Code d'erreur (0=Ok)
|                BX=Nombre de pages logiques attribuées
|
|      Le tableau fait 2 word : le handle puis le nbre de pages.
|      L'EMS étant limité à 256 handles, le tableau ne peut dépasser
|      le 1 KO.
```

Voilà les services de l'EMM détaillés. Vous retrouverez tout ceci sous forme de code dans les annexes.

Codes d'erreurs du driver EMS

Squirrel [Aou 95]

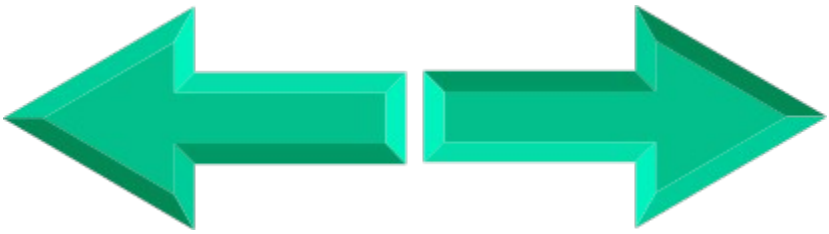
Codes d'erreurs de l'EMM.

```
00 = Opération réussie
80 = Erreur Interne, EMM détruit
81 = Electronique EMS défaillante
82 = EMM occupé
83 = Handle incorrect
84 = Fonction non reconnue
85 = Plus d'handle disponibles
86 = Erreur de concordance de pages
87 = Dépassement de pages
88 = Mémoire insufisante
89 = Impossible d'allouer 0 page
8A = Numéro de page logique non valable
8B = Numéro de page physique non valable
8C = Cellule d'allocation saturée
8D = Handle déjà sauvegardé
8E = Handle non sauvegardé
8F = Sous-fonction non reconnue
```

Transmis dans AL, le reste des registres renvoyés n'est bien sûr interpretable que si AL=0.

Justifions-Nous ?

Stessy



Ce petit article sans prétentions va traiter de la justification d'un texte. On ne trouve guère de sources traitant de ce sujet, et lorsque le besoin s'en fait sentir il faut s'atteler soi-même à la tâche.

Ayant du en passer par là, je vous livre le résultat de mes propres cogitations. La méthode n'est guère compliquée d'ailleurs. Aussi me contenterais-je de vous la fournir sous forme d'algorithmes informels facilement transposables dans n'importe quel langage.

L'objectif est d'obtenir un texte précédé de n espaces correspondant à la marge gauche, et dont chaque ligne est de même longueur (intervalle entre les marges "droite" et "gauche"). Ceci s'obtient en répartissant des espaces entre les mots (je ne parle que du mode texte).

Par convention on part du principe que le texte à traiter, l'est en totalité et préalablement chargé en mémoire dans une liste chainée du type :

```
RECORD = TEXTE
                    Chaine : TABLEAU[0..255] DE CARACTERES
                    Succ   : POINTEUR
                    Pred   : POINTEUR
                FIN_RECORD
```

Il est également possible de travailler à partir d'un tableau de chaines, mais si cette fonction doit être utilisée au sein d'un autre logiciel celà devient restrictif. De plus l'algorithme s'en trouve compliqué.

Quelques variables sont déjà connues, comme le nombre de lignes à justifier (NBL), les marges voulues etc...

```
               NBL=1728                                          (* Nombre de
lignes à traiter           *)
               Marge_Gauche=xx                             (* Valeur donnée par
l'utilisateur *)
               Marge_Droite=xx                             (* Valeur donnée par
l'utilisateur *)
               Intervalle    =Marge_Droite-Marge_Gauche (* Largeur que devra
*)

(* avoir chaque ligne          *)
               Ptr_Dbt          =xx                          (* Pointeur sur la
1ere ligne           *)
               Ptr_Fin          =xx                          (* Pointeur sur la
dernière ligne    *)
               Ptr_Ici          =RIEN                      (* Pointeur de
travail                      *)
```

1. Première étape avant de justifier un texte : le déjustifier.
Ca peut paraitre contradictoire mais indispensable, vu qu'à priori la routine ne peut savoir si des espaces ont déjà été insérés lors d'une précédente justification du texte.

C'est le rôle de la routine "Dejustifier_Texte" du listing qui suit.

Cette routine va rechercher dans chaque ligne de texte les espaces doubles, (significatifs d'une justification) et supprimer l'un des deux. Cette opération est répétée pour une ligne donnée tant qu'il subsiste des espaces doubles. Au final on obtient un texte brut ou l'espace joue son rôle normal de séparateur de mots.

Remarquez que lorsqu'une ligne est traitée entièrement, juste avant de passer à la suivante on vérifie que le premier caractère n'est pas un espace. Si c'est le cas on le supprime. Cela facilitera la justification qui consiste à remplacer un espace unique par deux tant que la ligne n'a pas la longueur voulue. Si on laissait ce premier espace on créerait une marge droite aléatoire.

2. La seconde étape porte sur la justification à proprement dit.
Elle se décompose en trois traitements distincts.

- Le premier consiste à fusionner la chaine courante avec les suivantes, ce tant que la longueur de la chaine courante est inférieure à la valeur de Intervalle (Intervalle=Marge_Droite-Marge_Gauche)

- Le second dépendant du premier, consiste à couper cette chaine entre deux mots de manière à obtenir la plus longue chaine possible rentrant dans l'intervalle. La portion de chaine excédante redevient la chaine suivante.

- La troisième enfin consiste à remplacer chaque espace par un double espace tant que la longueur de la chaine n'est pas égale à l'intervalle. Puis remplacer les codes de contrôles par un espace et enfin recréer la marge gauche par l'insertion de n espaces (n=Marge_G) au début de la chaine.

Le principe est assez simple, mais c'est un peu plus compliqué à programmer (pas trop je vous rassure). Il y a juste quelques petits pièges à éviter :

- Si la ligne en cours est suivie d'une ligne blanche (saut de ligne) ou par une ligne commençant par un des caractères : -
  * ◆ (caractéristiques du début d'une énumération) on ne
  procède pas à la concaténation.

- Si la ligne en cours est la première d'une énumération (donc commençant par - * ◆) et que le second

caractère est un espace on le remplace par un code de controle (ascii 255 par exemple). Ceci permet de préserver un espace unique entre le - * ◆ et le premier mot de l'énumération. Par ailleurs, comme il est de règle de décaler une énumération, on fait précéder cette ligne deux deux codes de contrôles et on fixe une variable numérique (Retrait) à 1 après l'avoir justifiée. Les codes de contrôles servent uniquement à occuper une position dans la chaine courante avant son passage dans la routine de justification à proprement dit. Ensuite ils sont remplacés par des espaces. La variable Retrait est remise à 0 lorsque la ligne en cours est l'item suivant d'une énumération (donc ligne commençant par ◆ - *) ou une ligne blanche.

- Si la variable Retrait est à 1 et que la ligne courante ne commence pas par - * ◆ on l'incrémente pour indiquer qu'il s'agit de lignes faisant partie d'une énumération et subis- sant donc un retrait de paragraphe.

- La dernière ligne d'un paragraphe n'est pas forcément à justifier, celà dépend de sa longueur. L'exemple qui suit procède à sa justification seulement si sa longueur avant justification est supérieur à 2/3 d'intervalle.

- Si la variable Retrait est = 2 on la fait précéder de 4 caractères de contrôle avant de procéder à sa justification. On préserve ainsi l'alignement de l'énumération.

Il est bien entendu possible d'enrichir cet algorithme de base. Par exemple pour qu'il tienne comptes des énumérations numériques du style 1) 1. 1- etc...

Le jeu se complique un peu plus si l'on souhaite faire une justification avec césure en fin de mots. L'algo présenté se contente de scinder les lignes entre deux mots. Il est possible d'y adjoindre la gestion des césures à condition de programmer un analyseur syllabique qui permettra de déterminer avec justesse le point de césure. Il conviendra également d'y ajouter dans ce cas une routine de contrôle afin que le nombre de césures par paragraphe ne soit pas trop excessif.

Pour terminer je vous conseillerais si vous avez eu quelque mal à comprendre le présent article, de procéder à la petite expérience suivante :

- Dans un éditeur de texte quelconque, chargez un fichier non justifié. Ceci fait essayez de le justifier manuellement de la manière la plus rationnelle possible. Logiquement vous devriez rapidement déceler les mécanismes que doit mettre en oeuvre un algorithme de justification pour arriver à ses fins. C'est d'ailleurs ainsi, en formalisant des actions manuelles sur un texte, que j'ai élaboré la routine que je vous livre ici.

Stéph'

Et maintenant, le listing :

```
    (*------------------QUELQUES VARIABLES UTILES---------------------*)

    NBL=1728                                                  (* Nombre de lignes à
traiter            *)
    Marge_Gauche=xx                                    (* Valeur donnée par
l'utilisateur *)
    Marge_Droite=xx                                    (* Valeur donnée par
l'utilisateur *)
    Intervalle    =Marge_Droite-Marge_Gauche (* Largeur que devra avoir      *)
                                                                              (*
chaque ligne *)
    Ptr_Dbt          =xx                                  (* Pointeur sur la 1ere
ligne            *)
    Ptr_Fin          =xx                                  (* Pointeur sur la
dernière ligne    *)
    Ptr_Ici          =RIEN                              (* Pointeur de travail
*)
    Ptr_Tmp          =RIEN                              (* Pointeur provisoire
*)
    ChaineTmp      =""                                  (* Chaine provisoire
*)
    Index              =0                                    (* Compteur
*)
    Nb_To_Insert=0                                     (* Nombre d'espace à insérer
pour    *)
                                                              (* qu'une ligne
soit justifiée          *)
    Insert_Ok      =FAUX                               (* Flag déterminant si on peu
justi*)
                                                              (* la chaine
*)
    Sens_GD          =FAUX                              (* Permet de justifier de
la gauche*)
                                                              (* vers la
droite et inversement      *)
                                                              (* pour
équilibrer l'insertion des *)
                                                              (* espaces
*)

    (*---------------------LA ROUTINE PRINCIPALE----------------------*)
    APPELE_ROUTINE Dejustifier_Texte
    Ptr_Ici=Ptr_Dbt
    BOUCLE
        APPELE_ROUTINE Joindre_Chaines
        APPELE_ROUTINE Scinder_Chaine_Courante
        SI (LONGUEUR(Ptr_Ici^.Succ^.Chaine)>0) ET
              (LONGUEUR(Ptr_Ici^.Chaine>0)) ALORS
              APPELE_ROUTINE Justifier_Chaine_Courante
        AUTREMENT
              SI LONGUEUR(Ptr_Ici^.Chaine)>((Intervalle DIV 3)*2.5) ALORS
                    APPELE_ROUTINE Justifier_Chaine_Courante
              FIN_SI
        FIN_SI
        SI Ptr_Ici=Ptr_Fin ALORS
              QUITTE_BOUCLE
        AUTREMENT
              Ptr_Ici=Ptr_Ici^.Succ
        FIN_SI
```

```
    FIN_BOUCLE
    (*------------------ROUTINE Dejustifier_Texte---------------------*)
    ROUTINE Dejustifier_Texte
    Ptr_Ici=Ptr_Dbt
    BOUCLE
        BOUCLE
              Pos_Dbl_Espace=CHERCHE_DANS_CHAINE("    ",Ptr_Ici^.Chaine);
              SI Pos_Dbl_Espace<LONGUEUR(Ptr_Ici^.Chaine) ALORS
                    EFFACE_DANS_CHAINE(Ptr_Ici^.Chaine,Pos_Dbl_Espace,1)
              AUTREMENT
                    SI Ptr_Ici^.Chaine[0]=" " ALORS
                          EFFACE_DANS_CHAINE(Ptr_Ici^.Chaine,0,1)
                    FIN_SI
                    QUITTE_BOUCLE
              FIN_SI
        FIN_BOUCLE
        SI Ptr_Ici DIFFERENT_DE Ptr_Fin ALORS
              Ptr_Ici=Ptr_Ici^.Succ
        AUTREMENT
              QUITTE_BOUCLE
        FIN_SI
    FIN_BOUCLE
    FIN_ROUTINE Dejustifier_Texte
    (*------------------ROUTINE Joindre_Chaines ----------------------*)
    ROUTINE Joindre_Chaines
    SI Ptr_Ici DIFFERENT_DE Ptr_Fin ALORS
          SI LONGUEUR(Ptr_Ici^.Chaine)>0 ALORS
                BOUCLE
                    SI (LONGUEUR(Ptr_Ici^.Chaine)>Intervalle) OU
                          (Ptr_Ici=Ptr_Fin) ALORS
                          QUITTE_BOUCLE
                    FIN_SI
                    SI (LONGUEUR(Ptr_Ici^.Succ^.Chaine)>0) ALORS
                          SI CHERCHE_DANS_CHAINE(Ptr_Ici^.Succ^.Chaine[0],"-*◆")<2
ALORS
                                QUITTE_BOUCLE
                          AUTREMENT
                                CONCATENATE(Ptr_Ici^.Chaine," ",ChaineTmp)

CONCATENATE(ChaineTmp,Ptr_Ici^.Succ^.Chaine,ChaineTmp)
                                OldPtrSuccSucc=Ptr_Ici^.Succ^.Succ
                                SI Ptr_Ici^.Succ=Ptr_Fin ALORS Ptr_Fin=RIEN FIN_SI
                                DE_ALLOUE(Ptr_Ici^.Succ)
                                Ptr_Ici^.Chaine=ChaineTmp
                                Ptr_Ici^.Succ      =OldPtrSuccSucc
                                OldPtrSuccSucc^.Pred=Ptr_Ici
                                SI Ptr_Fin=RIEN ALORS Ptr_Fin=Ptr_Ici
                          FIN_SI
                    AUTREMENT
                          QUITTE_BOUCLE
                    FIN_SI
                FIN_BOUCLE
```

```
          AUTREMENT
                Retrait=0
          FIN_SI
    FIN_SI
    SI Ptr_Ici^.Chaine[0]=" " ALORS
          EFFACE_DANS_CHAINE(Ptr_Ici^.Chaine,0,1)
    FIN_SI
    (* Ici on traite le cas ou la ligne est la première d'une énumération    *)
    (* ou un ligne faisant partie d'une énumération
*)
    SI CHERCHE_DANS_CHAINE(Ptr_Ici^.Succ^.Chaine[0],"-*◆")<2 ALORS
          INSERE(ASCII(255),Ptr_Ici^.Chaine[0])
          INSERE(ASCII(255),Ptr_Ici^.Chaine[0])
          SI Ptr_Ici^.Chaine[3]=" " THEN Ptr_Ici^.Chaine[3]=ASCII(255);FIN_SI
          Retrait=1;
    AUTREMENT SI Retrait>0 ALORS
          INSERE(ASCII(255),Ptr_Ici^.Chaine[0])
          INSERE(ASCII(255),Ptr_Ici^.Chaine[0])
          INSERE(ASCII(255),Ptr_Ici^.Chaine[0])
          INSERE(ASCII(255),Ptr_Ici^.Chaine[0])
          Retrait=2;
    AUTREMENT
          Retrait=0;
    FIN_SI
    FIN_ROUTINE Joindre_Chaines
    (*------------------ROUTINE Scinder_Chaine_Courante----------------*)
    ROUTINE Scinder_Chaine_Courante
    SI LONGUEUR(Ptr_Ici^.Chaine)>Intervalle ALORS
          Index=Intervalle-1
          BOUCLE
                  SI Ptr_Ici^.Chaine[Index]=" " ALORS QUITTE_BOUCLE FIN_SI
                  SI Index>0 ALORS
                        DECREMENTE(Index)
                  AUTREMENT
                        Index=Intervalle-1    (* Cas ou la chaine traité ne contient
*)
                        QUITTE_BOUCLE              (* aucun espace ou faire la
scission      *)
                  FIN_SI
          FIN_BOUCLE
          COPIE(Ptr_Ici^.Chaine,ChaineTmp)
          EFFACE_DANS_CHAINE(Ptr_Ici^.Chaine,Index,Fin_Chaine)
          EFFACE_DANS_CHAINE(ChaineTmp,0,Index)
          ALLOUE(Ptr_Tmp,TEXTE)
          Ptr_Tmp^.Succ=Ptr_Ici^.Succ
          Ptr_Tmp^.Pred=Ptr_Ici
          Ptr_Tmp^.Succ^.Pred=Ptr_Tmp
          Ptr_Ici^.Succ=Ptr_Tmp
          SI Ptr_Ici=Ptr_Fin ALORS Ptr_Fin=Ptr_Tmp FIN_SI
          Ptr_Tmp^.Chaine=ChaineTmp
    FIN_SI
    FIN_ROUTINE Scinder_Chaine_Courante
    (*------------------ROUTINE Justifier_Chaine_Courante--------------*)
    ROUTINE Justifier_Chaine_Courante
    AVEC_BASE_REFERENCE Ptr_Ici
        LgnChaine=LONGUEUR(Chaine)
        Nb_To_Insert=Intervalle-LgnChaine
        Insert_Ok=FAUX
        Index=0
        BOUCLE
              SI Nb_To_Insert=0 ALORS QUITTE_BOUCLE;FIN_SI
```

```
      SI Sens_GD ALORS
            SI (Index>=LgnChaine) ET
                  (Insert_Ok=FAUX) ALORS QUITTE_BOUCLE;FIN_SI
            SI (Index>=LgnChaine) ET (Insert_Ok=VRAI) ALORS
                  Index=LgnChaine-1
                  Sens_GD=FAUX
            AUTREMENT
                  SI (Chaine[Index]=" ") ET (Nb_To_Insert>0) ALORS
                        INSERE(" ",Chaine[Index])
                        INCREMENTE(LgnChaine)
                        DECREMENTE(Nb_To_Insert)
                        Insert_Ok=VRAI
                        BOUCLE
                            SI Chaine[Index]=" " ALORS
                                  INCREMENTE(Index)
                            AUTREMENT
                                  QUITTE_BOUCLE
                            FIN_SI
                            SI (Index>=LgnChaine) ALORS QUITTE_BOUCLE; FIN_SI
                        FIN_BOUCLE
                  AUTREMENT
                        INCREMENTE(Index);
                  FIN_SI
            FIN_SI
      AUTREMENT
            SI (Index=0) ET (Insert_Ok=FAUX) ALORS QUITTE_BOUCLE;FIN_SI
            SI (Chaine[Index]=" ") ET (Nb_To_Insert>0) ALORS
                  INSERE(" ",Chaine[Index])
                  INCREMENTE(LgnChaine)
                  DECREMENTE(Nb_To_Insert)
                  Insert_Ok=VRAI
                  BOUCLE
                      SI Chaine[Index]=" " ALORS
                            DECREMENTE(Index)
                      AUTREMENT
                            QUITTE_BOUCLE
                      FIN_SI
                      SI (Index=0) ALORS QUITTE_BOUCLE;FIN_SI
                  FIN_BOUCLE
            AUTREMENT_SI Nb_To_Insert=0 ALORS
                  QUITTE_BOUCLE
            AUTREMENT
                  DECREMENTE(Index)
            FIN_SI
            SI (Index=0) ET (Insert_Ok=VRAI) ALORS
                  Index=1
                  Sens_GD=VRAI
            FIN_SI
      FIN_SI
FIN_BOUCLE
(* La ligne courante étant justifiée on va rechercher d'éventuels    *)
(* codes de contrôles que l'on aurait insérés et les remplacer par *)
(* des espaces *)
BOUCLE
      Pos_Code_Controle=CHERCHE_DANS_CHAINE(ASCII(255),Chaine)
      SI Pos_Code_Controle<LONGUEUR(Chaine) ALORS
            Chaine[Pos_Code_Controle]=" "
      AUTREMENT
            QUITTE_BOUCLE
      FIN_SI
FIN_BOUCLE
```

```
        (* Il ne reste plus qu'à recréer une marge gauche pour la la ligne *)
        (* courante en insérant n espace avant avec n=Marge_G
*)
        POUR Index=1 ALLER_A Marge_G
            INSERE(" ",Chaine[0])
        FIN_POUR
    FIN_AVEC_BASE_REFERENCE
    FIN_ROUTINE Justifier_Chaine_Courante
    (*-------------------------------------------------------------------*)
```

Le Concours du Reporter

Les premières froidures de l'automne arrivent. C'est l'époque où les programmeurs, même les plus sociaux, voient avec joie le terme de la traditionnelle sortie annuelle familiale. Celle que le commun des mortels nomme vacances, mais qu'entre eux ils désignent plus volontier sous le terme de :

"Procédure de traitement préventif du syndrome YaKeTonPcKiCompte".

Au revoir donc les savantes équations tracées dans le sable, dont le voisin de plage prend un malin plaisir à fausser le résultat en écrasant son mégot entre deux chiffres, laissant accroire qu'il y a une virgule là ou il n'y en avait pas...

Au revoir les kilomètres de rallonges électriques et téléphoniques que l'on se coltine du bungalow loué à la plage, faute de ne pouvoir se payer de modem satellitaire et dix accus de rechange pour le portable...

Terminée la longue et quotidienne séance d'enduit à la spatule de crème solaire indice 27 Maxi-Super Ecran total. Besogne indispensable si l'on veut garder son teint d'endive légèrement jaunatre, jalousement peaufiné à longueur d'année à la lueur de vagues reflets cathodiques (imaginez la tête des "collègues" programmeurs célibataires si l'on revenait plus noir qu'un fond d'écran du dos:-<).

LES VACANCES SONT FINIES, PLACE A LA VRAIE DETENTE....

Première tâche du programmeur rendu à son habitat naturel, (s'il n'a pu trimballer avec lui 10 km de rallonges diverses) : consulter son courrier électronique, et s'enquérir des nouvelles et nouveautés...

"Ah Oh tiens tiens....Que vois-je ? Le Rep' 8 est sorti ? Que ne voilà-t'y pas de la saine lecture pour me remettre dans le bain" *(bain de pseudo radiations émises par le moniteur bien sûr, pas bain de mer, faut suivre...)*

"Voyons voir....Quoi d'vieux, ouais boff... Tiens - Le Concours du Reporter- Quesaquo ?"

"...Les premières froidures...blabla...blabla...mais on parle de moi là..."

"...tiens ils savent même que je pense en moi-même «mais on parle de moi là».."

EXIT; (<- sortie précipitée d'un article récursif sans condition)

* * *

Et bien oui cher lecteur, il nous arrive de traiter l'information sur le vif, en temps réel... Mais à propos d'information, as-tu retenu le titre de cet article ?

"Le concours du Reporter", c'est celà.

Ce que tu ne sais sans doute pas encore, c'est que pendant que tu cherchais à résoudre le délicat problème opposant :

- l'usage d'un clavier,
- à la conjugaison de doigts poisseux de crême solaire,
  sur lequels viennent s'agglutiner des grains de sables,

pendant ce temps disais-je, le Reporter No 7 est paru.

Si tu l'eus su, que tu l'eus eu, et que tu l'eus lu, tu ne te serais sûrement pas demandé "Le concours du Reporter, Quesako?" (voir plus haut), n'est-il pas ? comme diraient nos cousins anglo-saxons.

Sache Oh vénérable lecteur qu'à l'initiative de Legolas, un concours de programmation, gratuit, accessible à tous quelque soit son niveau (enfin presque),et doté de cadeaux pour les 50 premiers participants (sans parler du/des gagnant(s)) à été ouvert dans nos colonnes....

Il serait dommage que tu n'y participas point. Et quelle meilleure remise en condition qu'un petit concours de programmation après de longues semaines d'abstinences concédées à la diplomatie familiale ?

Si tu veux en savoir plus, hâte-toi d'aller quérir le No 7 du Reporter chez ton fournisseur habituel.

Quand bien même d'ailleurs, le concours ne t'interresserait pas, il serait dommage de laisser ta collection du Reporter orpheline d'un si beau numéro (il paraitrait qu'il porte chance, si,si...)

Toute l'équipe te souhaites une excellente lecture, et attend avec impatience le résultat de tes C.A.O. (Cogitations Assistées par Ordinateur).