



64
no
ps

numero 2 : janvier 2024

Une histoire de la demoscene Amstrad CPC.

AMSTRAD DEMOSCENE, KESAKO ?

Dans quel pays le demomaking sur CPC est-il réellement né ? Quel a été le premier demomaker marquant de l'histoire ? Que se passait-il durant les meetings dans les années 1990 ? Pourquoi tout s'est-il effondré en 1993 ? Les codeurs de démos étaient-ils plus brillants que les codeurs de jeux ? Les conflits entre CPCistes ont-ils toujours existé ? Quel a été le rôle des swappers et des fanzines dans l'évolution de la scène ? D'où vient la bisbille entre la France et l'Allemagne ? Pourquoi le CPC ne s'est-il jamais imposé sur la scène démo comme une machine de référence ?

Ce livre, qui couvre les années 1984 à 1997, est le premier ouvrage sur le sujet et vise à mieux faire connaître la demoscene Amstrad CPC et à rendre hommage aux CPCistes actifs durant cette période.

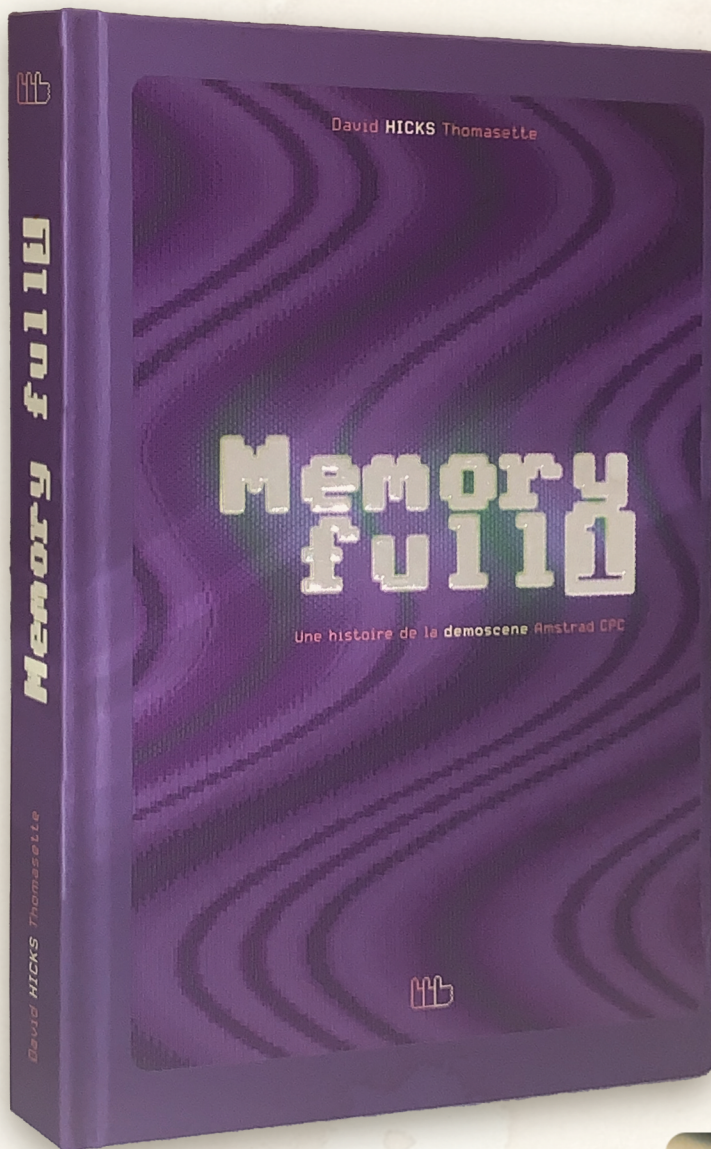
Logon System, Malibu Coders, BENG!, Paradox, UniX, Symbiosis, etc. On connaît parfois les grandes lignes de cette histoire, mais elle reste souvent stéréotypée. Ce livre cherche au contraire à rendre compte du phénomène au sens large, en évitant de le réduire aux démos références. Il se penche sur le rôle et l'importance des fanzines, des meetings, des swappers, ou des associations. En se replongeant dans l'ambiance de l'époque, on suit l'apparition des péripéties de la scène de façon chronologique, tout en proposant des analyses techniques accessibles afin de mieux mesurer la valeur des différentes démos.

"C'est un livre magnifique."

- H., graphiste

"Ça se lit tout seul."

- H.P., sorcier



SOMMAIRE

Chapitre 1 : les prémices (1984 – 1988).

Où l'on s'intéresse aux techniques des jeux qui anticipent celles des premières démos, à l'apparition des cracktros, et l'émergence du phénomène au Danemark et en Allemagne.

Chapitre 2 : l'âge d'or (1989 – 1993).

Où l'on décortique l'explosion du phénomène, qui atteint la France, la constitution de gros groupes et d'une véritable scène, l'apparition des fanzines et des meetings.

Chapitre 3 : le creux de la vague (1994 – 1997).

Où l'on suit l'évolution de la scène après la mort commerciale de la machine, qui entraîne un abandon massif d'une première génération de demomakers, et un renouveau difficile.

Glossaire.

Où les termes techniques utilisés dans le livre sont définis le plus simplement possible, afin de rendre la lecture accessible à tous.

CARACTÉRISTIQUES

180 pages couvrant la période 1984 à 1997.

213 images extraites des démos, jeux et logiciels les plus marquants de cette période.

Dimensions : 14,8 x 21 cm (hors couverture).

Couverture cartonnée pelliculage mat + vernis sélectif.

Impression offset couleur sur papier 130 g/m².

Toi aussi abonne-toi et lâche trois pouces si tu as aimé cette publicité.



ÉDITO

64 NOPs en 64 pages, vous tenez entre vos mains un magazine parfaitement synchronisé !

Fidèle à l'esprit du demomaking, ce second numéro de 64 NOPs s'ouvre avec un nouveau record : la plus grande image jamais réalisée dans une résolution CPC. Ce superbe mode 0 occupe en effet 8 fullscreens (384×1088 en 26 couleurs) ! Nous devons cette petite prouesse artistique à Made qui a eu la gentillesse de nous offrir cette magnifique couverture (un making of sera bientôt publié).

Cette fois-ci, l'équipe de rédaction s'est efforcée de proposer des articles plus accessibles que dans le précédent numéro, tout en conservant des pages s'adressant aux plus confirmés. Ainsi au fil de votre lecture, vous en apprendrez plus sur les projets en cours grâce au compte rendu de la Benediction Coding Party #3 et aux interviews de certains acteurs de la scène CPC. Une rubrique « graphisme » fait son apparition, ainsi qu'une rubrique « making of » pour y découvrir les secrets de fabrication des dernières démos. Bien entendu, les traditionnels articles consacrés au code sont toujours au rendez-vous.

Je tiens à remercier tout particulièrement Hwikaa pour son aide précieuse lors de la mise en page de ce numéro. Le magazine n'en est que plus beau.

Enfin, vous aurez sans doute remarqué l'apparition de quelques publicités. Celles-ci vont permettre à toute l'équipe de financer des vacances bien méritées jusqu'au numéro 3 !

Bonne lecture !

TOMS/PULPO CORROSIVO

4

LE MEETING DE LA BÉNÉDICTION

8

INTERVIEW : GHOSTS'N GOLEM13

14

DEMO IZ MAGIC!

18

LE A.B.B.A. DU CONTRASTE

22

LE COUP DU POULPE

26

INTERVIEW : ELIOT / BENEDICTION

31

LOADER BASIC BINAIRE

36

VIVE LE « CRTIC 1 ONLY » !

39

OPTIMISER SON BC26

43

SAMPLEUR ET SANS REPROCHE

48

DES TRIANGLES SUR CPC

54

**INTERVIEW :
PASCAL VIELHESCAZE**



LE MEETING DE LA BÉNÉDICTION

Clac ! C'est avec bien du mal que vous réussissez enfin à boucler votre inusable sac de randonnée, qui a vu bien des meetings... Vous voici fin prêt pour rejoindre Gavray-sur-Sienne, dans la Manche, où vous allez vivre 4 jours de la Benediction Coding Party #3 !

PAR... Vous !

1

Après 9 heures d'un paisible trajet, la magnifique salle des fêtes se dresse enfin devant vous. Elle est parfaitement adaptée à l'événement : spacieuse, dotée d'un bar, d'une grande cuisine, d'un espace nuit, et même d'un projecteur relié à un grand écran ! Des gîtes sont à proximité pour des nuits plus calmes et confortables. Après avoir installé votre matériel, vous apercevez un petit groupe rassemblé au bar. Si vous souhaitez le rejoindre, rendez-vous au 16, si vous préférez vous laisser attirer par les sons soundchip provenant du CPC situé derrière vous, rendez-vous au 5.

2

Vous décidez de rester fidèle à *Orgams*, votre X-MEM et votre Albireo. La journée s'achèvera par une soirée pizzas, grâce au savoir-faire de roudoudou (encore lui), Golem13, MvKTheBoss et Beb, tous mobilisés en cuisine pour l'occasion. Vous avez clairement trop mangé, mais ce n'est pas bien grave. Vous décidez de vous coucher tôt. Rendez-vous au 20.

3

Alors que vous vous dirigez avec détermination vers le cake au beurre trônant bien en évidence sur le



bar, vous tombez nez à nez avec Nicky One et Bouba, les deux anciens membres de DBT ! Vous êtes soudainement envahi par un élan de nostalgie : l'originalité de la *Sweet Megademo*, les délires de *Boxon 3*, les « digiprouts », le dresseur d'ours, la finale *Soul Almighty*... Toute votre adolescence remonte ! Les deux bougres gardent un œil sur le CPC depuis toutes ces années et sont à deux doigts de replonger. Afin de vous remettre de vos émotions, vous décidez de vous ruer sur le dernier paquet de M&M'S restant. Rendez-vous au 11.

4

Après deux jours sans douche, vous estimez qu'il est temps de prendre un peu soin de vous, car les autres participants commencent à vous regarder de travers. Ça tombe bien, Beb est en partance pour le gîte, situé non loin de la salle, où Genesis8 dort encore. Vous décidez de faire le trajet avec Beb, non sans une certaine appréhension. Vous découvrez après quelques pas un personnage fin et cultivé, capable de passer de l'évocation de l'expressionnisme abstrait de Mark Rothko à la dernière démo de TEK sur Amiga 500. Enfin douché, vous rentrez à la salle. Rendez-vous au 14.

5

Quelques CPCistes devisent autour de *DSC #4*, une démo de Longshot sortie deux jours avant le meeting. À peine arrivé, et vous voici déjà au cœur d'une riche discussion entre développeurs (émulateurs ou démos) autour de cet intéressant comportement du CRTC 1. Vous ne voyez pas le temps passer. Rendez-vous au 12.

6

Vous décidez d'aller prendre un bon café bien chaud du côté du bar. Il est 1h du matin, le début de nuit peut encore être un peu productif. Au loin, vous apercevez la team Sud Ouest qui s'affaire pour votre plus grand plaisir. OffseT œuvre pour que Nicky One reparte avec une configuration CPC stable et prête à l'emploi. Quelques routines seront échangées, et vous vous mettez à rêver d'une *Sweet Megademo 2*. Si vous souhaitez retourner à votre CPC pour essayer de coder malgré l'heure avancée, rendez-vous au 9, si vous préférez rejoindre le petit attroupement formé autour du CPC de AsT, rendez-vous au 21.



7

Une après-midi d'échanges et de code plus tard, roudoudou s'empare du micro pour présenter le portage de ACE sur PC. Il insiste d'emblée sur le caractère collectif de son travail, en remerciant bien sûr OffseT pour le cœur d'émulation, Sid pour la compilation sur Mac, Pulkomandy pour la version Haïku,

sans oublier ses échanges avec Kukulcan, Fredouille (*Caprice32*) ou Lone (*SugarBox*). Suite à cette présentation, si vous décidez de basculer tous vos dévs en cours sous ACE, rendez-vous au 17, sinon rendez-vous au 2.

8

Vous approchez timidement du coin Benediction, qui laisse tourner sur son CPC une preview de la démo *Stand Up!*, présentée ce soir pour la compétition. L'équipe a enfin mis au point un système de développement répondant aux attentes de tous les membres. Ainsi, Eliot, Krusty et Voxfreax peuvent modifier des projets simultanément et travailler facilement de concert. Un bel esprit d'équipe ! Vous préférez attendre la diffusion officielle de la démo pour l'observer dans les détails. Rendez-vous au 18.

9

Vous tentez, sans y croire, de vous replonger dans votre dernier source *Orgams*. À vos côtés, Targhan poursuit son développement d'*Arkos Tracker 3*, comme à chaque meeting. Vous essayez de l'inciter à plutôt se concentrer sur un nouveau projet de jeu, dont il a été secrètement question la veille au soir. En guise de réponse, celui-ci vous rit au nez. Vexé, vous décidez finalement d'aller vous coucher avec des idées plein la tête. Rendez-vous au 25.

10

Un petit coup de déo fera l'affaire. Cela vous permet de reprendre un café, qui accompagnera magnifiquement les trop nombreuses viennoiseries que vous avez décidé d'engouffrer, en compagnie de Supersly, attablé avec son Banania. Vous taillez le bout de gras pendant une bonne heure. Teopl et son amie se joignent au groupe. Venu de Serbie, c'est son premier meeting, bien qu'il soit déjà l'auteur de *Ludic Break The Loop*, un jeu codé à l'occasion du CPC Retro Dev 2019, avec l'aide de son frère aux graphismes et aux musiques. Tout cet investissement finit de vous réchauffer le cœur, à moins que ce ne soit le café... Rendez-vous au 14.

11

C'est l'heure de la présentation du livre *Memory Full* par Hicks. Posté derrière le bar, micro en main, celui-ci revient sur la question de la fiabilité des sources historiques, l'importance de recouper les témoignages, et de proposer un récit circonstancié plutôt qu'une suite d'anecdotes. S'ensuit une discussion collective. Vous décidez immédiatement d'acheter le livre et de faire un don conséquent à Hicks afin qu'il puisse écrire le tome 2 rapidement. C'est l'heure du repas. Si vous souhaitez vous installer entre Zik et Beb, rendez-vous au 22, si vous préférez la compagnie de Tom et Jerry et Targhan, rendez-vous au 15.

12

Vous n'avez pas encore eu le temps de vous installer devant votre CPC qu'il est déjà l'heure du dîner. Encore de riches discussions en perspective. Mais c'est décidé : ce soir, vous coderez ! La soirée se poursuit entre Targhan et Eliot, dans une atmosphère assez productive. Mais le trajet était éreintant, et vous finissez par aller vous allonger vers 3 heures. Rendez-vous au 27.

13

Vous réalisez rapidement que vous avez fait le mauvais choix. 10 heures de route vous attendent, avec un CTM sur les genoux et un roudoudou imitant l'accent italien durant tout le trajet. Cependant, la chaleur d'un TotO et d'un AsT, blottis contre vous à l'arrière vous reconforte définitivement. *Graaande* week-end !

14

Une petite balade dans les allées de la salle vous permet de constater que les CPCistes sont bien studieux ce matin. Fredouille tente de debugger *Caprice* en s'aidant des résultats d'autres émulateurs, TotO a apporté un exemplaire d'*Alcon 2020* qui tourne en boucle sur son CPC, Cracky, un cracker local de l'époque (à ne pas confondre avec l'ancien membre de Cocoon System et Mortel) s'efforce de stabiliser un code de rupture avec l'aide d'Eliot... Vous croisez sur votre chemin Toug et Wild de *Crack'n'Rom*, encore des anciens de passage ! Ils avaient apporté une planche du numéro 2 de leur fanzine *CPC Fanz BZH*, qui bénéficie de l'aide d'un maquettiste. Soudainement, Krusty vous interpelle. Si vous décidez de le rejoindre, rendez-vous au 8, si vous préférez faire semblant de ne rien entendre, rendez-vous au 26.

15

Discussion jeux ! Quels étaient les meilleurs jeux sur CPC, finalement ? Targhan et Tom et Jerry sont très bien placés pour en parler, étant à la fois développeurs et joueurs à leurs heures perdues. Certains classiques comme *Rick Dangerous* ou *Shinobi* font tout de suite l'unanimité, mais d'autres font débat. Tom et Jerry nous apprend à l'occasion qu'il envisage de porter un nouveau jeu Apple II, dans la lignée de sa désormais célèbre trilogie : *Le Crime du Parking*, *La Java du Privé*, et *Le Mur de Berlin va Sauter*. Targhan reste malheureusement mutique sur la question. Rendez-vous au 7.

16

C'est les retrouvailles pour les premiers participants déjà arrivés. AsT enchaîne les anecdotes truculentes pendant que vous débattiez gameplay et level design avec TotO. En 30 minutes, vous imaginez ensemble pas moins de 6 nouveaux projets de

jeux totalement réalisables sur CPC. Il ne reste plus qu'à les coder ! Rendez-vous au 12.

17

Après avoir déclaré votre nouvelle conversion au cross-dev, Beb vous gifle bruyamment. Vous vous écroulez au pied du grand écran, rendez-vous au 25.



18

C'est l'heure de la présentation des productions qui concourent à l'occasion du meeting, sur écran géant s'il vous plaît ! On se croirait à la Revision. Vous vous installez confortablement entre AsT et Sid pour partager vos impressions... et un peu de whisky. Une seule démo en lice, *Stand Up!* de Benediction, qui propose des variantes de plasmas très bien finalisées et parfois originales (noise, plans différentiels, logo en surimpression). Côté jeux, vous tombez littéralement sous le charme de *Mighty Castle Adventure* développé par Yogtze, bien qu'encore à l'état de preview. Vous n'aviez plus vu d'aussi beaux graphismes en mode 1 depuis longtemps. Mais Targhan fait involontairement bugger le jeu pendant la démonstration ! Cinq images, enfin, sont proposées, par rexbeng, Sim1, MacDeath, Logone et Kris. C'est ce dernier qui remportera le plus de voix lors du vote post-meeting. Le graphisme de Sim1 est un écran original conçu sur place en quelques jours, sa première réalisation sur CPC. Les trames sont travaillées, la composition est bien pensée, vous aimez ! Il est déjà tard, et le sommeil commence à se faire sentir. Si vous souhaitez vous éclipser derrière le rideau de l'estrade pour dormir, rendez-vous au 25, sinon rendez-vous au 6.

19

Vous passez le reste du meeting à enchaîner les parties de *Goldorak* sans lever le nez. Tout à coup Eliot vous tapote sur l'épaule pour vous signifier qu'il faut quitter la salle. Le meeting est déjà terminé, vous rentrez chez vous un peu penaud.

20

C'est déjà dimanche et vous n'avez pas vu le temps passer. À peine sorti de votre duvet, vous êtes saisi par une odeur familière... Les croissants de Tom et Jerry sont bien là ! Et ils ne sont pas seuls, puisque Golem13 et MvKTheBoss ont doublé la mise en apportant leur propre livraison. Malheureusement, une odeur nauséabonde s'échappe de votre t-shirt en plein petit-déjeuner. Lancez deux dés 6. Si vous faites moins de 8, rendez-vous au 4, sinon, rendez-vous au 10.

21

Vous vous frayez un chemin parmi les curieux et découvrez sur le CTM de AsT une preview de la séquelle de *The One*, un nouveau déluge d'écrans hardware techniques. Quelques finitions manquent malheureusement à l'appel pour que la prod puisse être présentée en catégorie démo au meeting. Vous appréciez néanmoins ! Vous ne vous ennuyez pas de la soirée et faites connaissance avec Corinne, la nouvelle recrue d'Impact, surnommée « la bûche en escarpins ». Délire Impact/Vanity de rigueur. Vous finissez par vous effondrer sur votre matelas, rendez-vous au 25.

24

En traversant la salle, vous êtes interpellé par un grand chevelu, concentré sur son Amstrad Plus. Mais c'est Zisquier ! Presque nouveau venu en meeting, sa passion pour le CPC est palpable et communicative. Sa version de *Goldorak* est de plus en plus riche, et il tente devant vous de figurer les trajectoires de ses « Golgoths », qu'il trace sur une feuille de papier à petits carreaux. Le pad vous fait de l'œil. Si vous souhaitez débiter une partie, rendez-vous au 19, sinon rendez-vous au 11.

25

Réveil difficile ce matin, pour ce dernier jour. Les ronflements porcins d'un des participants alliés au manque de confort vous rappellent que vos premiers meetings sont maintenant bien loin. Cela dit, vous avez passé un excellent week-end, et remerciez chaleureusement Eliot pour l'organisation. Un dernier tour de la salle pour dire au revoir, et c'est le moment de rentrer. Si vous souhaitez embarquer avec roudoudou, TotO et AsT, rendez-vous au 13, si vous préférez la quiétude du convoi d'OffseT et Zik, rendez-vous au 23.



22

La discussion dérive rapidement sur la différence entre musique tonale et atonale. Beb et Zik s'accordent sur le caractère naturel et universel de la première. Difficile de transposer la seconde sur CPC... Vous vous mettez à imaginer collectivement la première démo « art contemporain », avec monochrome en guise de graphismes, musique atonale et code aléatoire, façon surréalistes. Dans cette logique, les codes qui plantent peuvent être vus comme des œuvres éphémères ! Une fois avalée votre Danette chocolat, rendez-vous au 7.

23

Après avoir pris quelques jours de congé dans la famille de Zik et OffseT, vous rentrez tranquillement à Antibes avec la ferme intention de vous installer définitivement chez ce dernier. À vous les soirées astronomie et z80, petit veinard !

26

Vous tournez les talons discrètement, et vous dirigez dans la direction opposée, vers l'équipe de Crazy Piri. RedBug et son musicien Shaan œuvrent ensemble pour la finalisation d'un *Pac-Man* qui dort sur un disque dur depuis près de deux ans. Et surprise, ils réfléchissent à des idées pour une future démo ! Affaire à suivre. Rendez-vous au 18.

27

Vous émergez progressivement, réveillé par des voix et des mélodies familières... La nuit sur votre couche de fortune (un maigre matelas) a été étonnamment calme et reposante. Si vous souhaitez prendre tout de suite votre petit-déjeuner, rendez-vous au 3, si vous préférez vous diriger vers votre CPC, rendez-vous au 24.



GHOSTS'N GOLEM13

Tel un maître forgeron façonnant patiemment et minutieusement son glaive dans la fournaise de son atelier, Golem13 peaufine actuellement les derniers détails de *Ghosts'n Goblins* sur Amstrad Plus. Entre deux séances de martelage de son assembleur, il a accepté de répondre à mes questions : réactions à chaud.

PAR TOMS/PULPO CORROSIVO

Bonjour Golem13 ! Peux-tu te présenter pour les lecteurs qui ne te connaîtraient pas encore, ton parcours sur CPC et surtout, pourquoi Golem13 ? C'est pour te porter chance ?!

Salut, mon nom est Frédéric Poesy. Bientôt quinquas, je suis marié et père de 3 enfants. Je me suis surnommé Golem13 dès mon débarquement sur IRCnet en 1996. En cette période, je sortais du service militaire et je travaillais chez Renault Trucks. Un bon collègue m'appelait souvent par ce surnom, sans que je sache réellement pourquoi et ça nous faisait marrer. C'était une belle époque, légère et festive. Alors quand il a fallu choisir un pseudonyme, le choix était évident : moins de 9 caractères, simple, lisible, et porteur d'excellentes vibrations : il était parfait.

Vers 11 ans, j'ai entamé mon voyage dans le monde de la programmation en utilisant le langage BASIC sur mon Oric Atmos, un ordinateur que mon grand-père, un passionné d'informatique, m'avait transmis. Par la suite, quelques années plus tard, j'ai naturellement poursuivi ma passion en explorant l'Amstrad, cadeau de Noël si déterminant de 1987. Bien aidé par nos magazines de l'époque, *Hebdogiciel* et

Amstrad Cent Pour Cent, c'est à ce moment-là que je me suis lancé dans la programmation en assembleur. J'ai écrit de nombreux jeux, ce qui m'a amené à écrire des utilitaires dédiés. J'ai même tenté de me familiariser avec la création de démos, mais je n'ai jamais dépassé le stade du *proof of concept*.

Vécu comme une certaine heure de gloire, j'avoue avoir la fierté d'avoir été publié en novembre 1990 dans *Amstrad Cent Pour Cent*, avec mon petit *Molecularr*. Écrit en trois semaines, durant l'été 90,



Molecularr

j'avais décidé d'envoyer ma disquette. Quelques semaines plus tard, je reçus l'appel téléphonique d'Alain Massoumipour, connu sous le pseudonyme de Poum, ce qui a vraiment été un moment marquant pour moi. Cela représentait un certain accomplissement pour le gamin de 15 ans que j'étais. Les compliments et encouragements de feu Poum lors de nos discussions téléphoniques m'ont beaucoup apporté et influencé. Elles m'ont donné confiance en moi. Je tiens ici à rendre un hommage chaleureux à cet homme passionné, d'une extrême gentillesse, pour qui je conserverai toute ma vie respect et reconnaissance.

Tu travailles depuis 2016 sur une adaptation de *Ghosts'n Goblins* pour Amstrad Plus, peux-tu nous en parler ?

J'ai commencé à bosser sur *Ghosts'n Goblins* en février 2016. Après des années loin de l'Amstrad, j'y ai replongé quand j'ai montré à mes enfants la « console » de papa quand il avait leur âge. Finalement, ils regardaient cela comme nous regardions étant jeunes les télé noir et blanc. Moi, en revanche, j'étais de nouveau conquis. Un mois après commençait le projet.

Avant de parler de comment ça avance, je voulais partager comment l'idée du projet est née. Quand je parcourais mes vieilles disquettes début 2016, toutes les images de ma passion d'enfance ont refait surface dans ma tête. À force de relire de vieux codes et de réutiliser de vieux logiciels comme DAMS, la passion s'est ravivée. Assez rapidement, l'idée de me relancer dans l'aventure a germé. Et puis, pourquoi pas ?

Ayant toujours souhaité adapter ce jeu qui me faisait rêver quand j'étais gamin, c'était le moment de voir si, avec toutes mes années d'expérience et ma carrière de développeur, je pouvais réaliser ce rêve sur cette machine. Dès lors, après quelques vérifications techniques, les travaux ont commencé. Un mois plus tard, Thomas –Winner– Ferté a rejoint le projet pour s'occuper de tout l'aspect graphique. C'est aussi la période où Richard –TotO– Gatineau, par l'intermédiaire de CPCWiki m'a mis en contact avec Julien –Ixien– Riet, qui allait finalement prendre en charge la partie sonore du projet. Le cœur de l'équipe était au complet.

Quel est l'état d'avancement ?

En 2023, le jeu est techniquement presque achevé, bien qu'il manque encore un peu de code pour gérer les graphismes du nouveau design. En effet, il y a eu une première version avec Winner aux graphismes. Cette version a fait l'objet d'une sortie pu-

blique qui est toujours disponible sur CPCWiki, mais elle ne propose que le 1^{er} niveau. Aux alentours de 2019 - 2020, après des mois d'atermoiements, car ne pouvant plus consacrer de temps au projet, Thomas s'est résolu à quitter l'équipe en laissant pas mal de travaux à finir.

À la recherche d'un remplaçant motivé et prêt à prendre en charge le travail d'un autre, j'ai fait la connaissance de Mathieu, connu sous le pseudonyme Hwikaa (à prononcer « ouika »). Cependant, Mathieu avait une perspective différente sur le projet et aspirait à des objectifs bien plus audacieux. Après quelques discussions et essais, nous avons convenu de recommencer le travail à zéro et de repenser complètement l'aspect visuel du jeu. Nous avons opté pour l'idée de préserver le gameplay original tout en extrapolant vers un univers visuel original. L'équipe lui a alors donné carte blanche pour exprimer tout son talent.

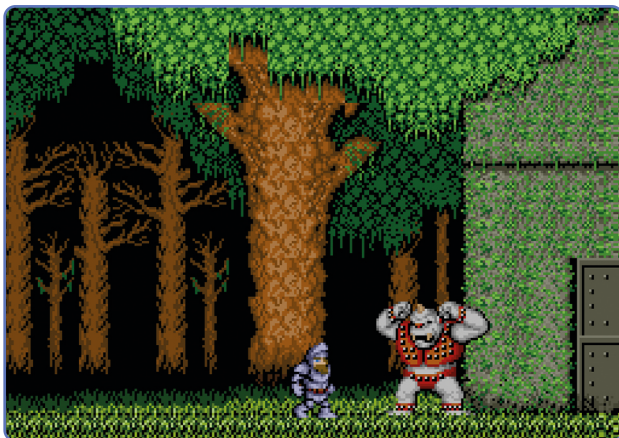


Golem13 et Hwikaa au Glop Meeting 3

Actuellement, il nous reste à compléter les graphismes des trois derniers niveaux, ainsi que quelques sprites. Je comprends que cela puisse sembler prendre du temps, car c'est effectivement le cas. Mais malgré les obstacles de la vie, je tiens à assurer à tout le monde que nous mènerons le projet à son terme.

Quels ont été les grands défis auxquels tu as dû faire face lors du développement ?

Quand j'étais gosse, j'ai énormément joué à *GnG* en salle d'arcade. Je le connais par cœur. J'ai d'abord commencé par passer en revue méthodiquement tous les points du jeu : les niveaux, les différents patterns, les tricks, le nombre d'ennemis à l'écran, etc. J'ai ensuite imaginé comment gérer techniquement chacun de ces éléments et déterminé leur faisabilité.



Ghosts'n Goblins sur borne d'arcade...



... et sur Amstrad Plus

Le premier défi a été d'analyser l'original de A à Z. Je suis passé ensuite aux différentes validations techniques, en termes d'affichage, de scrolling, de temps de traitement. Il a fallu déterminer avec certitude la faisabilité de chacun des passages. Ce n'est qu'une fois tous ces points clarifiés que le développement a débuté en tant que tel.

C'est ainsi qu'arrive le second défi : faire tourner et rentrer tout ça sur un Z80 cadencé à 3.33 MHz avec 128K de RAM et 512K de ROM. Pour tout un chacun, ça semble être des conditions royales. La barre ayant été mise si haut en termes d'exigence et de résultat, je vous assure que les contraintes de capacité machine sont énormes. Néanmoins, tout a été fait pour obtenir un *framerate* stable en toute circonstance, indispensable à une expérience de jeu réussie.

Ensuite, un autre grand défi a été le changement de graphiste. Heureusement, la collaboration avec Hwikaa est excellente. Son approche des travaux est globalement identique à la mienne : le souci absolu du détail et l'indéfectible envie de tendre vers une certaine perfection. Mathieu, en plus d'être un professionnel, est doté d'un don artistique et technique incroyable. Il réussit à faire des choses somptueuses, dont je n'aurais jamais espéré faire profiter *GnG*.

Graphiquement, nous avons procédé de la sorte : chaque proposition est précédée d'une réflexion et/ou discussion plus ou moins longue afin d'établir des pistes. Puis une maquette est produite, défendue, puis critiquée par les autres membres de l'équipe. Avec argumentation, ils acceptent, ajustent, rejettent les idées, voire en proposent d'autres. En outre, malgré leur apparence prometteuse lors de la phase de conception, de nombreuses idées ont été écartées une fois mises en œuvre et intégrées dans le jeu. Cela implique de défaire, refaire et de nouveau proposer, jusqu'à l'entière satisfaction de toute l'équipe.

Cette façon de fonctionner, transversale au développement, est coûteuse et prend énormément de temps. Néanmoins, elle garantit un résultat de qualité dont tout le monde est satisfait.

Est-ce que le choix de l'Amstrad Plus t'a semblé évident dès le départ ? Quelles caractéristiques de la machine exploites-tu ? Utilises-tu des techniques hardware pour repousser certaines limites imposées par la machine ?

L'idée de départ était de se baser sur un Amstrad CPC 6128 stock (dit Old, sans extensions). Dans ma jeunesse, je n'étais pas familier avec le 6128 Plus, si ce n'est à travers les articles d'*Amstrad Cent Pour Cent*. Toutefois, suite à trois semaines d'investigations et de validations techniques, l'idée d'exploiter la gamme Plus s'est imposée. Le 6128 Plus, modèle phare et navire amiral, correspondait exactement à ma philosophie : utiliser la machine standard la plus puissante de la marque afin de concevoir la meilleure adaptation possible.

Forcément, j'ai voulu exploiter toutes les capacités offertes par la machine. Parmi elles : le port cartouche et ses 512K de ROM, les sprites hard, le scrolling hard précis au pixel et à la ligne, la possibilité d'utiliser 32 encres parmi 4096 couleurs, réparties en 2 palettes de 16 encres chacune, les DMA pour interagir avec la puce sonore, et pour finir l'utilisation complète des 128K de RAM. J'ai ajouté la gestion de la PlayCity de TotO, permettant de jouer les musiques sur le PSG interne, tout en disposant les SFX en son spatial sur les deux PSG ajoutés par la carte.

Afin de compléter les 16 sprites hard, qui étaient insuffisants pour reproduire fidèlement le gameplay original, j'ai choisi d'implémenter une gestion complète de sprites soft, affichés et découpés au pixel. Ils se marient en toute transparence à l'écran grâce à une utilisation astucieuse des palettes. Cette déci-

sion a nécessité l'utilisation d'un double tampon (double buffer) pour gérer les affichages de manière efficace. Les animations s'exécutent de manière fluide à une fréquence constante de 25 Hz, quel que soit ce qui est à l'écran.

Afin d'obtenir des temps de traitement efficaces lors de l'affichage des sprites – qu'ils soient matériels ou logiciels – j'ai recouru à la technique de compilation de sprites sous toutes ses formes. Il s'agit de code natif exécuté pour afficher la majorité des éléments, toutes les fois où c'est possible. En contrepartie, cette méthode requiert beaucoup de mémoire pour y dérouler le code.

Pour résumer, toutes les capacités de la machine sont mises en œuvre, dans les règles de l'art, sans « trick ». La grande partie de la puissance réside dans la capacité du code à les exploiter au maximum, avec souplesse et élégance.

Tu as essayé d'être autant fidèle que possible au jeu original. Ce manque de liberté n'est-il pas trop frustrant ? Tu n'as pas été tenté de créer des niveaux bonus inédits ?

À mon avis, la source de frustration découle principalement de l'adaptation initiale réalisée par Elite en 1985. Malgré sa qualité compte tenu des contraintes de l'époque et du temps de développement limité (quelques semaines), elle laissait place à des possibilités d'amélioration. Par conséquent, dès le commencement du projet, notre objectif principal était de créer une conversion presque parfaite, en incorporant cette fois-ci tous les éléments emblématiques du jeu.



Ghosts'n Goblins sur CPC (1985)

Dans le domaine de l'adaptation la plus fidèle possible, le cahier des charges est déjà fixé. Le manque de flexibilité ne suscite pas tant de frustration que de véritables défis techniques. La difficulté réside dans la recherche de solutions techniques appropriées

pour respecter ce cahier des charges, en particulier lorsque la machine cible est nettement moins puissante que la machine d'arcade originale. Cette approche diffère de celle de la création d'un jeu original, où chaque élément peut être défini en fonction des capacités techniques de la machine destinataire. Dans le cas de *GnG*, la situation est à l'opposé.



Ghosts'n Goblins sur Amstrad Plus

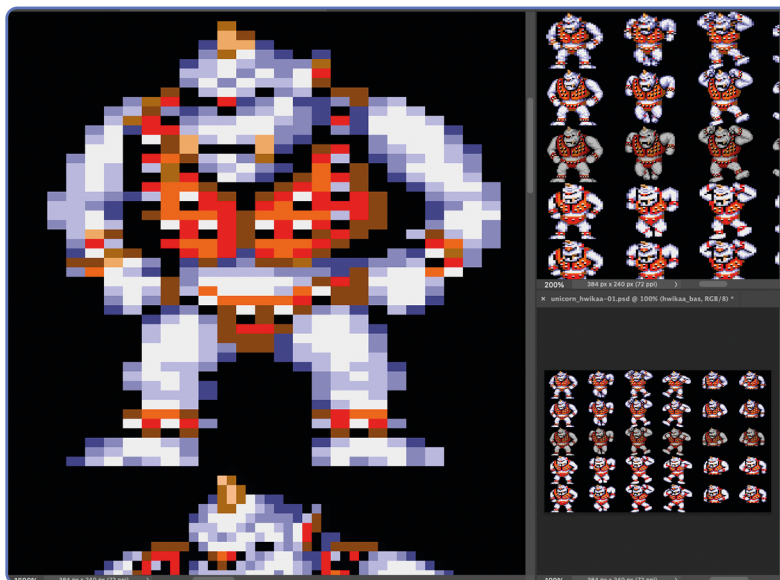
En fin de compte, je considère qu'avoir un cahier des charges clairement défini à suivre est source d'une grande satisfaction pour moi. Cela me permet de me focaliser sur ce que j'apprécie le plus : résoudre des problèmes techniques en réponse à des contraintes spécifiques.

La suggestion de concevoir des niveaux cachés revient très fréquemment. Pour tout bon jeu, cela s'explique par le fait que l'expérience est si plaisante que l'on souhaite la prolonger. Pour être franc, bien que ce soit une très bonne idée, je ne pense pas que nous allons allouer de l'espace sur la cartouche à quelque chose que peu de gens verront. Nous préférons réserver ces ressources précieuses pour garantir une réalisation de qualité.

Cependant, dans une approche un peu différente, cette idée pourrait donner naissance à un tout nouveau jeu *Ghosts'n Goblins* qui reprendrait les niveaux, l'atmosphère et le bestiaire que nous connaissons, mais avec un tout nouveau level design. Cela donnerait une nouvelle vie à ce jeu emblématique. À ce stade, l'idée est encore en phase de gestation, mais plus j'y réfléchis, plus je trouve que l'effort serait à la fois efficace et gratifiant. Je suis sûr que nous aurons l'occasion d'en reparler.

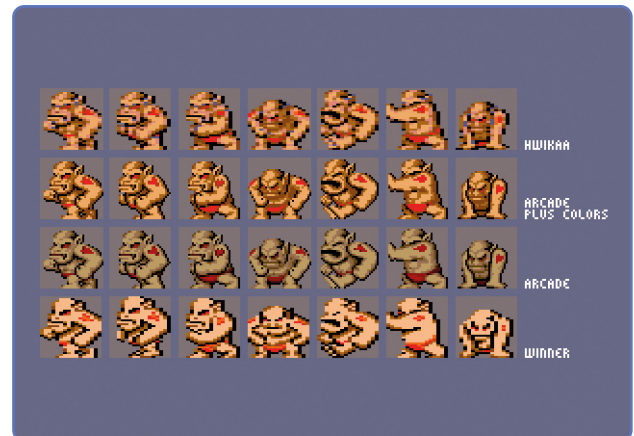
Quels outils as-tu utilisés tout au long du développement ? Est-ce que ton expérience de développeur professionnel t'a été utile ?

Étant développeur de métier, je me suis spécialisé dans le domaine de l'informatique de gestion et j'ai travaillé dans le secteur financier au Luxembourg pendant près de deux décennies. Ma spécialisation se concentre sur l'écosystème de développement Microsoft. Cette expertise a indéniablement joué un rôle crucial dans mon approche du projet. Elle m'a permis d'anticiper les problèmes, de prendre des décisions éclairées en amont et de rectifier rapidement mes erreurs. De plus, en étant spécialisé en C# .NET, j'ai pu mettre au point les outils nécessaires pour la création du jeu, que ce soit des compilateurs, des outils de traitement des données, des compresseurs, ou même des logiciels d'édition visuelle pour les graphistes. Ils ont tous été développés en C#.



Ghosts'n Goblins 128K est totalement écrit en assembleur. L'assembleur, le langage, se présente sous une forme assez simple : du code dans des fichiers texte, qui sont interprétés par un assembleur, le logiciel, pour produire un code binaire exécutable. Pour éditer de tels fichiers, j'utilise *NotePad++*. Il s'avère léger, portable, flexible, gratuit, et offre une coloration syntaxique du code. Il s'est révélé être un outil idéal. Pour les assembler, le meilleur et le plus puissant outil du marché est indéniablement *rasm* d'Édouard –roudoudou– Bergé, qui s'intègre parfaitement dans la plateforme de développement.

Car dans le contexte d'un jeu, en plus de l'assembleur, il y a également toute la gestion des actifs et des données de niveaux qui orchestrent le déroulement du jeu. Ces éléments nécessitent une gestion,



une préparation et une intégration complexes. Tous les processus impliqués, depuis la préparation des actifs jusqu'au traitement des données et à l'assemblage final, ont été entièrement automatisés grâce à un ensemble d'outils développés en C#. Sans la nécessité d'une intervention manuelle, il faut moins de 10 secondes pour générer le fichier cartouche final prêt à l'emploi. Cette automatisation a grandement simplifié les tâches fastidieuses et répétitives souvent associées au processus de développement, rendant ainsi indolores de nombreuses étapes qui n'ont plus à être effectuées manuellement.

Hwika utilise *Photoshop* pour créer les graphismes et fournit les planches au format GIF. Elles sont consommées par la plateforme. La conception des niveaux est effectuée en utilisant un logiciel que j'ai développé en C#. Cette application offre une interface graphique permettant de disposer les tuiles des différents niveaux tout en

prenant en compte les contraintes techniques du jeu, telles que les palettes et leurs changements, le partage de tileset, etc. Cet outil produit un fichier XML contenant toutes les données essentielles, qui est ensuite utilisé par la plateforme.

Ixien a initialement commencé son travail avec *Starkos*, le tracker CPC développé par Targhan. En cours de projet, nous avons fait le choix de passer à *Arkos Tracker 2*, une nouvelle version du même auteur, qui venait de voir le jour. Cette transition nous a permis de bénéficier des améliorations de cette version tout en travaillant sous l'environnement Windows. J'ai réécrit le lecteur *Arkos* pour le jeu, pour lui donner la capacité de gérer à la fois la musique et les effets sonores, avec compatibilité avec la PlayCity de TotO, tout en utilisant les DMA de la gamme Plus.

Ainsi, nous disposons désormais d'une plateforme solide, complète et constamment en évolution pour soutenir notre développement.

À titre personnel, quels sont les atouts qu'un jeu se doit de posséder pour que tu apprécies d'y jouer ?

Tout d'abord, je suis d'avis qu'un jeu doit offrir un défi constant. L'utilisateur doit toujours avoir le sentiment que le jeu ne le punit pas arbitrairement, mais qu'il le récompense pour ses efforts. En conséquence, le joueur doit percevoir qu'avec un peu d'entraînement, il peut s'améliorer continuellement.

Un excellent jeu doit être doté d'un moteur de jeu précis et réactif qui répond immédiatement à chaque interaction de l'utilisateur, sans aucun délai ni erreur perceptible. L'utilisateur doit se sentir en totale maîtrise de ses actions.

Cependant, si en plus de cela, le jeu peut offrir une expérience immersive, un gameplay fluide exécuté avec finesse, des graphismes attrayants et une musique envoûtante, alors cela ne peut être que bénéfique.

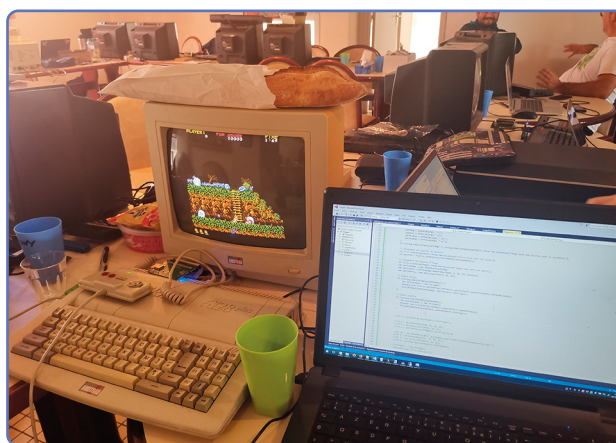
C'est précisément ce que nous nous efforçons de réaliser avec *Ghosts'n Goblins*. J'espère sincèrement que cela plaira aux joueurs.

Je sais que tu es peu intéressé par le demomaking, une raison particulière à cela ? Quel est ton regard sur la scène démo ?

Sur le plan technique, ce que j'apprécie particulièrement dans ce domaine, c'est souvent l'apparente complexité des algorithmes utilisés, pour découvrir

des astuces ingénieuses aux résultats époustouflants. Ce genre de chose m'a toujours impressionné. Sur le plan artistique, j'apprécie les travaux fluides et propres, avec des transitions soignées, sans à-coup visuel ou musical. On retrouvera ici mon affection pour les détails.

Néanmoins, je dois avouer que les démos n'ont jamais vraiment été ma tasse de thé, c'est vrai. Je peux les apprécier, mais je ne me sens pas capable d'en créer. J'ai bien essayé un peu quand j'étais plus jeune, mais cela n'a jamais dépassé le stade du POC. Les démos d'aujourd'hui sont bien au-delà de ce que j'ai pu faire à l'époque. Cela met la barre très haut, à laquelle tout nouvel arrivant est confronté. Je ne m'y sens pas.



En plein test à la Benediction Coding Party #2

La raison principale est que, par rapport au déroulement statique d'une démo, aussi maîtrisée et impressionnante soit-elle sur le plan technique, j'apprécie davantage la gestion sensible et dynamique de l'assemblage des cycles de vie simultanés de multiples éléments. J'apprécie l'idée de permettre à un joueur, véritable variable d'incertitude, d'évoluer et de perturber l'évolution de cet univers partagé, borné et maîtrisé, et d'interagir avec celui-ci.

Merci beaucoup pour le temps que tu nous as consacré. Comme le veut la tradition, si tu souhaites faire une annonce, passer un bonjour ou pousser un coup de gueule, le mot de la fin te revient...

Merci à toi. Que chacun puisse s'épanouir au mieux en pratiquant sa passion autour de notre machine de cœur. La bise !





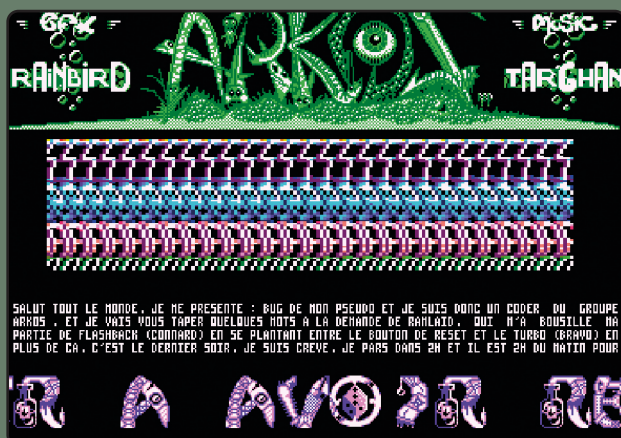
DEMO IZ MAGIC!

Pour commencer modestement cette série d'articles décortiquant des techniques utilisées dans les démos, je vous propose de nous attaquer à un effet de la *Batman Forever* : la barre mappée. Nos outils : des moulinettes en BASIC, de l'assembleur Z80, un cerveau et une cafetière.

PAR MAGIC ELIOT/BENEDICTION



Batman Forever (Batman Group)

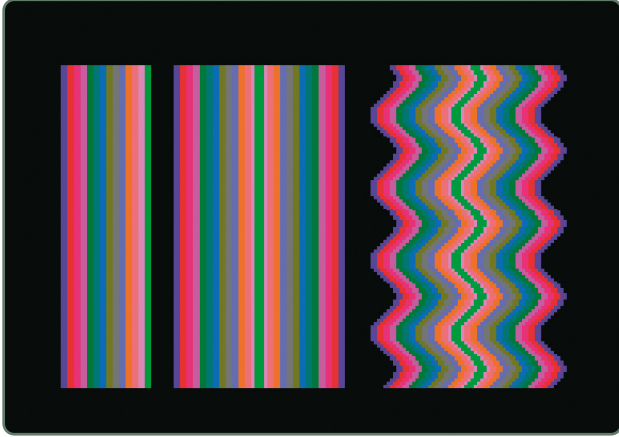


Bordelik 3 (Arkos)

Illustrant une fois de plus que l'imagination peut être la clé de nouveaux effets impressionnants pour pas cher, cette barre mappée repose sur 2 techniques bien connues et très abordables : un déplacement vertical ici réalisé avec un simple changement d'offset (R12 et R13) et un changement d'une grande partie de la palette toutes les 4 lignes, les magic rasters ! Ces fameux magic rasters ont été nommés et mis en œuvre pour la première fois par Orphée et Rainbird dans la démo *Bordelik 3* d'Arkos (3^e partie) en 1993. Le très amusant et atypique jeu *Trailblazer* sorti en 1986 utilisait déjà une technique similaire en changeant 5 couleurs par ligne.

Le principe des magic rasters repose sur un changement de palette rapide sur un dessin astucieusement établi que nous appellerons support. Rien n'est donc affiché, la texture est posée sur le support ! Notez que, puisque nous modifions la palette régulièrement, nous avons donc la possibilité de créer des textures utilisant jusqu'à 27 couleurs simultanément !

Pour rentrer dans le vif du sujet, nous allons afficher une texture de 14 couleurs par ligne sur 3 supports simples générés par le programme BASIC SUPPORT.BAS. C'est là qu'intervient la cafetière : vous faire patienter pendant la génération de l'image ! Bien sûr, votre graphiste attiré se fera un plaisir d'être plus imaginaire que des courbes sinus.



Comme tout codeur le sait depuis 1984, l'adressage d'une couleur sur nos CPC se fait en 2 temps via le Gate Array (voir l'article *Les rasters* sur le *Quasar Net* par exemple).

```
ld bc,&7F00+numéro_encre
out (c),c
ld a,couleur
out (c),a
```

L'opération sera simplement répétée 14 fois de suite dans une boucle, pour 14 encres différentes, en pointant sur une table de couleurs avec la pile (LD SP, TABLE_MAGIC) et en récupérant dans HL une valeur 16 bits, soit 2 couleurs d'un coup, avec l'instruction POP HL. Tout ceci en 3 NOPs !

Voici la structure de la table MRHAT.TAB générée par le programme CAPTURE.BAS à partir de 5 fichiers .SCR ayant chacun leur palette et composant une grande texture en 27 couleurs dessinée par Voxfreax/Benediction. Nous perdons volontairement 2 octets (00 00) pour que chaque ligne de magic rasters tiennent sur 16 octets, ce qui nous arrangera par la suite...

```
3000 56 46 44 55 54 44 44 54 54 5C 5C 54 5C 54 00 00
3010 40 53 57 55 54 54 54 54 5C 5C 58 5C 58 5C 00 00
3020 55 57 57 55 54 54 54 5C 5C 45 5C 45 5C 4C 00 00
```

Dans la boucle MAGIC_LOOP, nous allons lire ces données et changer les 14 couleurs à la suite :

```
MAGIC_SIMPLE
di
ld (PILE+1),sp
ld e,HAUTEUR
ld b,&7F ; adressage du GATE ARRAY

; La pile pointe sur
; la table de magic rasters

ld SP, TABLE_MAGIC

MAGIC_LOOP
ld d,1 ; pen = 1

pop hl ; récupère 2 couleurs
out (c),d ; sélectionne encre 1
out (c),l ; envoie la couleur
inc d ; passe à l'encre suivante
out (c),d ; sélectionne encre 2
out (c),h ; envoie la couleur

pop hl ; récupère 2 autres couleurs
inc d ; passe à l'encre suivante
out (c),d ; sélectionne encre 3
out (c),l ; envoie la couleur
inc d ; passe à l'encre suivante
out (c),d ; sélectionne encre 4
out (c),h ; envoie la couleur

[IDEM pour 5,6,7,8,9,10,11,12]

pop hl
inc d ; 13
out (c),d
out (c),l
inc d ; 14
out (c),d
out (c),h

; Lecture des 2 octets vides
; pour alignement à 16

pop hl

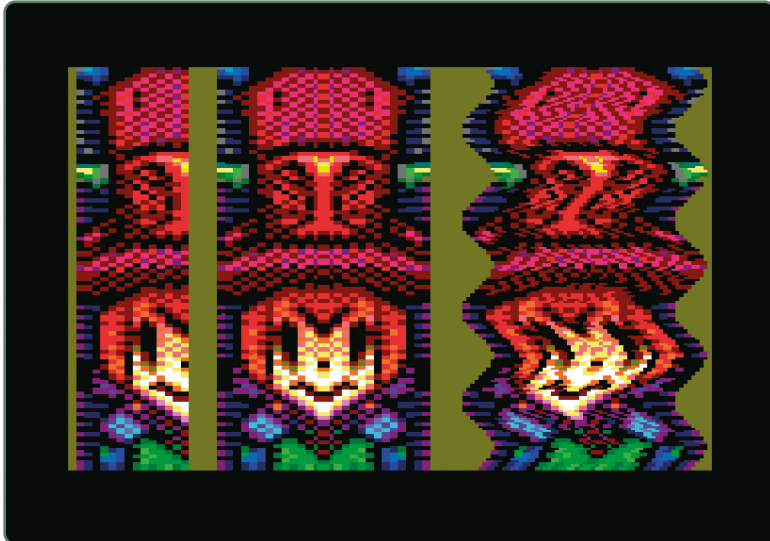
; Décompte de HAUTEUR puis on boucle
; tant que E n'est pas égal à 0

dec e
jr nz, MAGIC_LOOP

; On récupère la valeur
; de la pile sauvée avant la boucle

PILE ld sp,0
ei
ret
```


Avec le source EXEMPLE1.ASM, nous obtenons donc cet écran fixe, avec près de 27 couleurs, qui je le sens, vous donne déjà plein d'idées ! Notez que notre texture est appliquée sur plusieurs supports différents sans aucun effort supplémentaire et permet donc de faire un effet miroir ou des répétitions gratuitement !



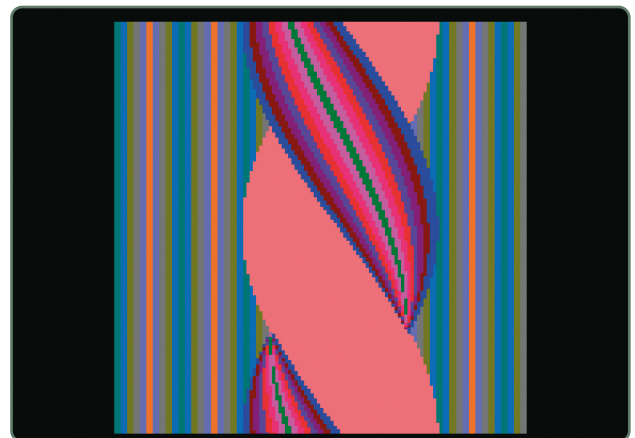
La boucle dure 155 NOPs, soit environ 2,5 lignes à l'écran, une ligne durant 64 NOPs chez tout codeur normalement constitué. Les plus observateurs auront remarqué sur leur moniteur que les 3 dessins sont à certains endroits légèrement différents sur une même ligne : ceci s'explique par le temps nécessaire pour changer une encre ! D'autre part, certains se poseront la question de savoir s'il est gênant de ne pas se caler sur un nombre entier de lignes pour changer la même encre. Je suis d'avis que non, au contraire, ça participe à un lissage de l'ensemble plutôt que de trouver les légers décrochés toujours au même endroit verticalement ! Les demomakers C64 ne s'embêtent pas avec ce genre de considérations...

À peu de choses près, il s'agit de la méthode utilisée par Rhino sur la barre mappée qui nous intéresse. Techniquement, il y a moyen d'envoyer plus rapidement une palette de couleurs, parfois avec des contraintes : OUTI, OUT (&FF), A... Voir l'article de toms dans le premier numéro de 64 NOPs. À présent, nous pouvons gérer le déplacement du pointeur de la table des couleurs, chargé dans SP, pour faire bouger la texture de haut en bas. Une classique lecture de table auto-bouclante de 256 octets (TABLE_SINUS) suffira pour se déplacer dans la texture tous les 16 octets (comme par hasard...) et révéler le potentiel de cette technique. Ceci est dans le source EXEMPLE2.ASM. Notez que le code de cette « démo » tient en moins de 256 octets et que l'animation est évidemment à 50 Hz...

Dans notre boucle, nous pointons jusqu'à maintenant 1 seule table de magic rasters, mais pourquoi ne pas gérer plusieurs tables qui seraient lues indépendamment les unes des autres ? Nous allons créer un nouveau support avec, au premier plan, une rubber bar de 8 encres en magic rasters et un fond de 6 encres en magic rasters. Ce support est généré par la grosse moulinette SUPPORT3.BAS, l'écran a été reformaté (64 octets sur 256 lignes, soit R1 = 32 et R6 = 32 du point de vue CRTC) pour faciliter le scrolling vertical à venir et avoir un motif bouclant.

Au niveau de notre boucle, en complément de la table de rasters de la barre (MAGIC_TABLE8) pointée par SP nous allons devoir intégrer la gestion d'un second pointeur pour se déplacer dans la table MAGIC_TABLE6, la texture du fond et envoyer 6 couleurs au Gate Array. Cela peut se faire par différents moyens, plus ou moins rapides : IX, registres secondaires (EXX), réutilisation de SP à condition de l'avoir sauvegardé... Cela s'optimisera en fonction de

ce qui sera fait en même temps que les magic rasters. Le code proposé ici est simple : la seconde table, pointée par HL, est lue par un classique LD A, (HL) et une fois la lecture des 6 couleurs terminée, HL est sauvegardé pour la prochaine itération avec LD (POINTEUR_TABLE6+1), HL.



Par clarté, j'ai laissé les INC HL qui pourraient être optimisés par des INC L pour 7 d'entre eux, car le dépassement de L ne se fera de toute façon que sur le 8^e INC HL. La boucle fait maintenant 180 NOPs, soit un peu moins de 3 lignes. Exécutons EXEMPLE3.ASM. Nous obtenons bien nos 2 plans indépendants en magic rasters. Il serait tout à fait possible de gérer encore plus de plans différents, en réduisant évidemment le nombre d'encres utilisées pour chaque plan...



Pour finir, dans EXEMPLE4.ASM, nous allons intégrer la gestion du scrolling hard vertical mettant en oeuvre une rupture 2 écrans et une complémentarité de leur registre 5. Le sujet, un peu technique, est déjà abordé dans le numéro 48 d'*Amstrad Cent Pour Cent* et sur *Quasar Net*. Cela dépasse le thème principal de cet article, je vous laisse étudier ça par vous-même ou alors opter, comme Rhino, pour un défilement vertical sans rupture, astucieux mais avec des contraintes.

Tous les fichiers mentionnés dans cet article sont disponibles à cette adresse sur le serveur de 64 NOPs :

64nops.memoryfull.net/files/issue-2/demo-iz-magic

```
inc d      ; le fond commence à l'encre 9

POINTEUR_TABLE6
ld hl, TABLE_MAGIC6
ld a, (hl) ; lit la couleur
out (c), d
out (c), a

inc d      ; passe à l'encre 10
inc hl     ; ou inc 1
ld a, (hl)
out (c), d
out (c), a

[IDEM pour 11,12 et 13]

inc d      ; passe à l'encre 14
inc hl     ; ou inc 1
ld a, (hl)
out (c), d
out (c), a

inc hl     ; ou inc 1

; On lit 2 valeurs inutiles dans la table
; car chaque ligne est stockée sur 8 octets

inc hl     ; ou inc 1
inc hl

; Sauve le pointeur HL car il sera modifié
; par un POP HL prochainement...

ld (POINTEUR_TABLE6+1), hl

; Décompte de HAUTEUR et rebelote...

dec e
```

Vous voici donc prêts pour réaliser de superbes effets avec une technique très abordable mais finalement peu exploitée depuis 1993, je dirais même sous-exploitée ! Les demomakers se seraient-ils réfrénés car « ce ne sont que des rasters » ? Les exemples proposés ici ne révèlent que 6 % du potentiel et il ne tient qu'à vous et votre designer de faire plus fin, par exemple. Les optimisations et autres possibilités sont nombreuses, certaines ont commencé à être explorées, d'autres non, en voici quelques-unes :

- dérouler la boucle
- lire une table avec un décalage de façon à créer une ondulation en X
- gérer plus de 2 plans de magic rasters
- modifier la table de couleurs pour animer la texture
- ajouter un délai, possiblement variable, entre le changement des encres afin de créer un effet de zoom ou poser la texture sur un support avec du relief : scroll *Star Wars*, boule mappée...
- insérer un changement d'offset grâce à une rupture pour avoir un support animé : rubber bar...
- exploiter la technique sur Plus avec un adressage de couleurs plus rapide grâce à l'ASIC et pourquoi pas dans les sprites hard
- échanger avec un designer/graphiste qui aura plein d'idées...

À bientôt dans votre prochaine démo ! ■

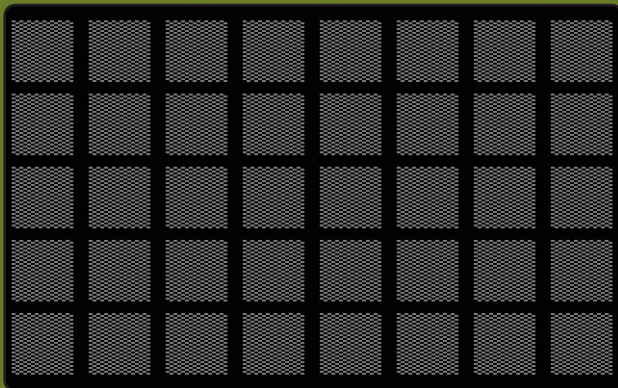


LE A.B.B.A. DU CONTRASTE

Toute composition visuelle raconte quelque chose. Il peut s'agir d'un message informatif sur un flyer, d'un aperçu de l'intrigue sur la couverture d'un livre, d'un événement épique sur une toile de maître, ou des conséquences d'un affrontement meurtrier dans la crypte d'un manoir isolé dans un jeu d'aventure sur CPC. Afin de transmettre nos idées, faire passer notre message, raconter notre histoire, en tant que créateurs graphiques nous voulons pouvoir diriger l'œil du spectateur dans notre image. Pour ce faire, les principes de la composition picturale mettent à notre disposition un certain nombre d'outils dont le contraste, qui est probablement un des plus efficaces.

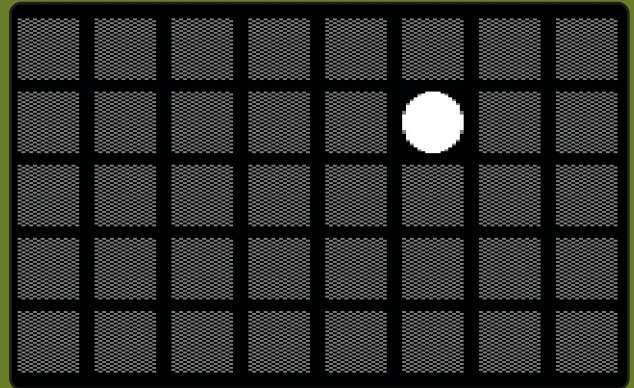
PAR HWIKAA/PRALINE (WWW.INSTAGRAM.COM/HWIKAA)

Pour illustrer la chose (c'est un minimum quand on parle graphisme), regardez cette grille :



Elle est parfaitement régulière, et si je voulais essayer de deviner quelle partie de cette image vous regardez, je serais bien en peine d'y parvenir. Avec un peu de bol (environ 2,5 % de chances), je pourrais tomber dessus par hasard.

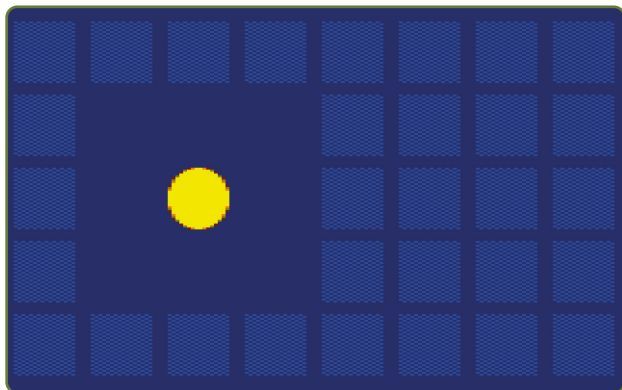
En revanche, si je vous demande de regarder cette image-ci :



Alors là, je suis certain que votre attention est attirée par le disque blanc. Et c'était exactement mon intention. Pour réaliser ce tour de magie, je n'ai fait qu'utiliser les super pouvoirs du **contraste**. Le contraste, c'est la mise en opposition de deux éléments qui va permettre de mettre l'un en exergue par rapport à l'autre. En vérité, j'ai même utilisé ici deux types de contraste à la fois, histoire d'être bien sûr de capter votre regard : un contraste de **valeur** (ou de luminosité), et un contraste de **forme**. Mon

disque est non seulement la seule forme circulaire au milieu d'un ensemble de carrés, mais il est également l'élément le plus lumineux d'une image majoritairement sombre : c'est à cet endroit que le contraste est le plus élevé, ce qui en fait le **point focal** de la composition.

J'aurais pu combiner ces petites astuces avec deux tricks supplémentaires en vous proposant cette image :



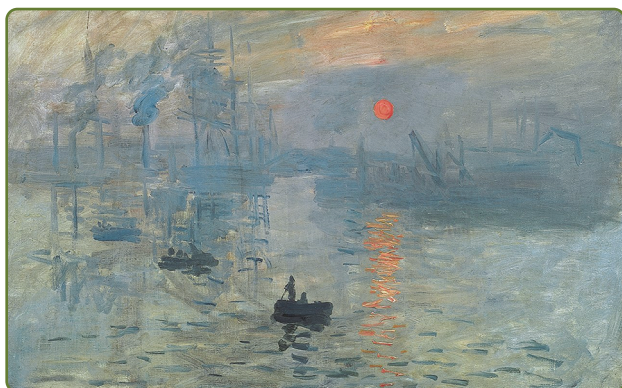
J'y ai ajouté un contraste de **couleurs** et j'ai isolé ma forme pour la détacher encore plus du reste.

Certes, ces formes sont plutôt abstraites, mais il se trouve que les grands peintres (enfin, je dis ça, mais si ça se trouve ils étaient loin de toucher mon plafond avec le sommet de leur crâne) usaient et usent encore des même techniques dans leurs compositions. Elles sont seulement mises en œuvre de manière un poil plus subtile.

Encore que.

MONEY, MONEY, MONEY

Regardez cette reproduction du tableau *Impression, soleil levant* de Claude Monet (ca 1872) :



On peut identifier deux points d'intérêt sur cette toile, entre lesquels notre regard se promène lorsque nous l'observons : la barque au premier plan et le soleil levant qui donne son titre à la peinture. Trois types de

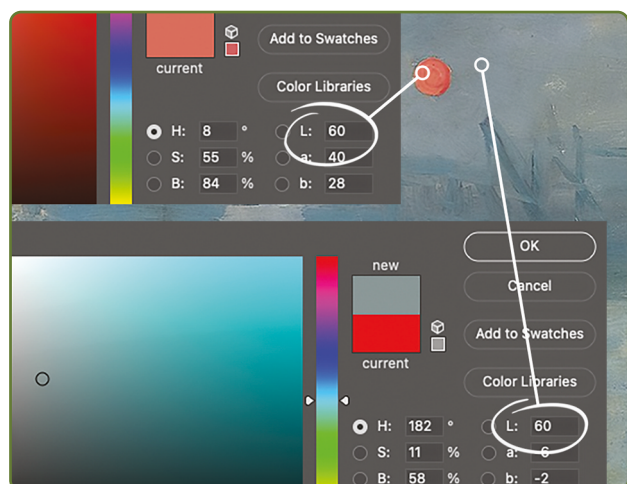
contrastes sont ici à l'œuvre : un contraste de valeur, un contraste de détail, et un contraste de couleur.

Si vous plissez les yeux, ou si je retire les données chromatiques du tableau, vous voyez que la barque se détache particulièrement du reste de la composition : c'est à cet endroit que se trouve la plus grande différence de luminosité, et cela attire inévitablement notre œil.

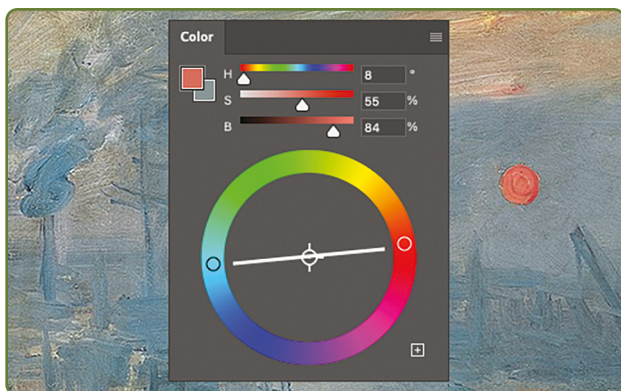


La barque est également l'élément qui présente les contours les plus nets, les plus définis ; tout le reste du tableau, y compris la seconde barque qui n'est pourtant pas si éloignée, est globalement flou. Il s'agit là encore d'un contraste visuel qui met l'accent sur l'embarcation.

Enfin, vous aurez probablement remarqué que le soleil, qui est pourtant particulièrement visible dans l'original, se fond dans le ciel sur cette version achromatique de la toile. C'est parce que sa teinte rouge-orangée et le turquoise du ciel ont la même valeur lumineuse :



On peut voir sur cette capture d'écran (aidez-vous d'une loupe si nécessaire) que les deux couleurs ont une valeur lumineuse de 60 (0 étant le noir le plus profond et 100 le blanc le plus brillant).



Or, il est intéressant de constater que ces deux teintes sont exactement diamétralement opposées sur le cercle chromatique, ce qui correspond à la définition des **couleurs complémentaires**, dont l'usage juxtaposé est une autre forme de contraste : c'est ce qui permet à ce soleil de tellement se démarquer sur un canevas majoritairement bleuté.

Une infinité d'autres œuvres utilisent ces techniques pour guider le regard ou mettre certaines parties d'une image en avant.



Ainsi, sur cet *Autoportrait avec palette* (1889), Vincent van Gogh reproduit exactement les mêmes procédés : contrastes de luminosité, de détail et de couleur permettent de focaliser l'attention sur le visage du peintre.

THE NAME OF THE GAME

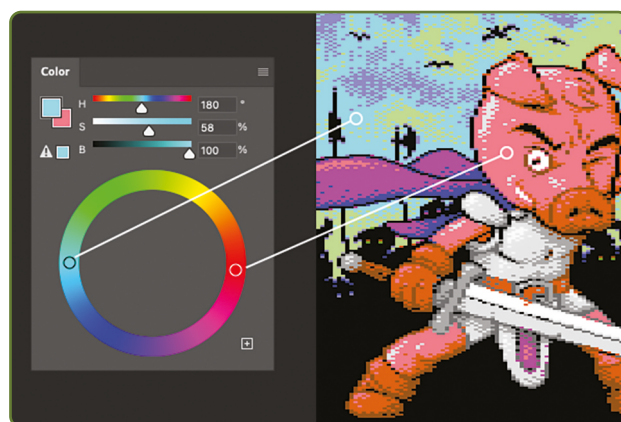
Je vous vais venir, vous allez me dire « OK, super, Hwika, mais on n'est pas tous des peintres flamands de la Renaissance » (Monet et van Gogh

non plus, hein, je vous ferai dire). Et puis on est dans un magazine qui traite du CPC, ne l'oublions pas. Qu'à cela ne tienne, allons voir ce qu'on peut trouver du côté... du C64. ^^

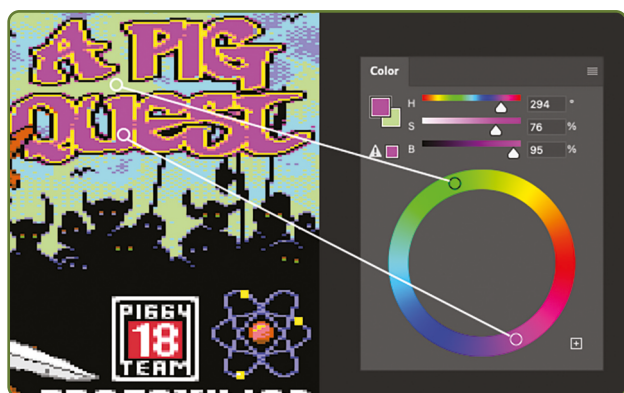
Mon crush vidéoludique du moment, c'est *A Pig Quest*, d'Antonio Savona et Mauricet.



Son écran-titre est une belle démonstration de mise en exergue par le contraste : tout est fait pour focaliser l'attention sur le personnage de Frank. Lequel, par ailleurs, pointe du doigt le titre du jeu. Mais nous parlerons lignes directrices dans un prochain article. Pour le moment, concentrons-nous sur les contrastes. Pour le détacher d'une foule compacte, le personnage est isolé au premier plan. Les ennemis sont uniquement suggérés par des silhouettes de casques à cornes et d'armes brandies, aucun n'est particulièrement reconnaissable, ils se valent tous, quand Frank est, lui, unique, net et bien détaillé. Enfin, l'armée n'est qu'une masse globalement noire alors que notre héros porcine est vif et coloré. Et puisqu'on parle de couleurs, j'attire votre attention sur le choix de ces dernières qui, comme vous allez le constater, est indéniablement délibéré.

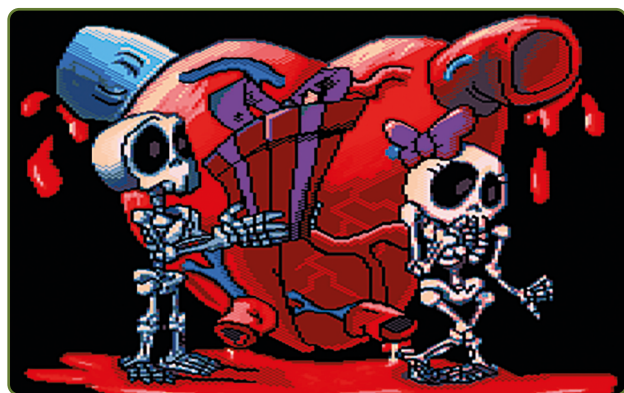


Voyez comme la teinte chair de la peau est diamétralement opposée au cyan du ciel ? Nous avons une fois de plus affaire à l'utilisation de couleurs complémentaires pour détacher le sujet du fond. Et la même astuce est utilisée dans la zone du titre.



LAY ALL YOUR LOVE ON ME

Passons, si vous le voulez bien, à un cas pratique. Il y a environ deux ans, Kris/iMPACT (merci à lui) a partagé cette création en mode 0 pour Amstrad Plus.



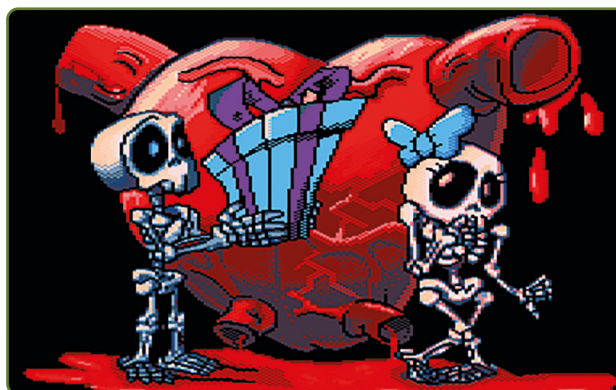
Après lui en avoir demandé l'autorisation, je me suis permis de lui adresser une critique constructive qui disait peu ou prou ce qui suit.

L'illustration usant majoritairement de tons rouges, l'introduction de la complémentaire (cyan) est une excellente idée pour guider l'œil du spectateur. Mais du coup, ici, on a surtout tendance à être attiré par les veines et la grosse artère sur la gauche. Et j'ai le sentiment que ce n'était pas le but de cette illustration.

Quand je la regarde, j' imagine que l'histoire qu'elle raconte est la suivante : *un squelette amoureux offre un cadeau à sa valentine*. De fait, j'aurais tendance à vouloir mettre l'accent sur les personnages et surtout sur le cadeau, plutôt que sur l'anatomie du cœur, et renforcer l'interaction entre les squelettes.

Pour ce faire, je retirerais tous les tons cyans del corazón et je les placerais sur le cadeau et le nœud de Miss Skeleton. L'image étant globalement rouge, l'attention se portera immédiatement sur les accents bleutés. J'utiliserais le gris violacé, qu'on trouve par exemple dans l'artère de droite, pour assombrir le bas du cœur et ainsi le rendre un peu moins présent visuellement, faisant ressortir davantage les personnages au premier plan. Et pour finir, je jouerais avec

les yeux des squelettes en leur donnant plus de présence et en les faisant se rencontrer, afin d'accentuer l'interaction entre les amoureux. Ce qui donnerait quelque chose comme ça, à la truelle :



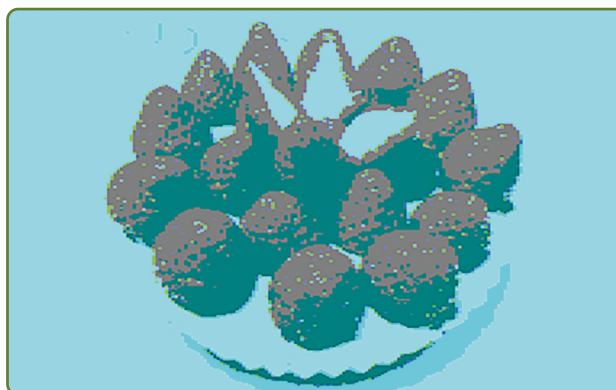
En ne faisant que redistribuer quelques couleurs, on a grandement amélioré la lisibilité de l'image et, surtout, **clarifié son message**.

So Long

Voilà qui conclut cette première approche de l'utilisation du contraste dans une œuvre picturale, quels que soient sa nature ou son support. Ces principes sont évidemment également applicables en architecture, dans une œuvre animée, et dans les jeux vidéo ! Il y aurait encore tellement plus à dire, à expliquer et à disséquer, mais je pense que je me suis suffisamment étalé pour aujourd'hui.

En complément, n'hésitez pas à aller lire mon article sur les notions de poids et d'équilibre visuels sur 64nops.wordpress.com (en anglais).

Et en attendant un prochain sujet, et pour rester sur le thème de la manipulation par la couleur, je vous laisse avec cette image intrigante du psychologue Akiyoshi Kitaoka, que j'ai adaptée sur CPC...



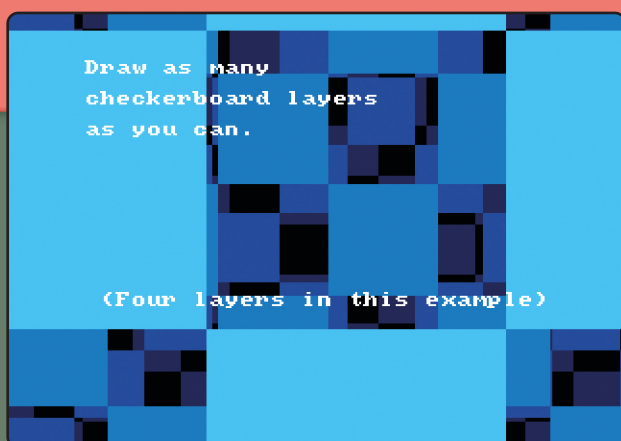
... ainsi qu'une question qui va potentiellement vous retourner le cerveau : me croyez-vous si je vous dis qu'il n'y a pas le moindre rouge dans cette image, et que ces fraises sont en réalité grises ? :) ■



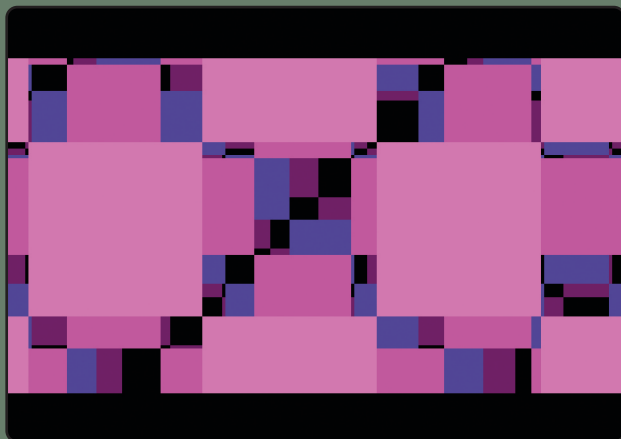
LE COUP DU POULPE

Petit guide pratique pour mater la concurrence en 4 coups.

PAR TOMS/PULPO CORROSIVO



Checkerboard Challenge (Loonies)



Checkmate (Pulpo Corrosivo)

Effet ultra-classique sur Amiga, les damiers immersifs sur plusieurs plans sont rares sur les machines 8 bits. Et une fois de plus, le CPC est à la traîne puisque jusqu'à présent il était possible de voir cet effet uniquement dans la démo *Can Robots Take Control?* de Benediction (sortie en 2021) dans laquelle Krusty réussit à afficher 3 plans.

Un checkerboard challenge sur Amiga avait été initié en 2017 par Blueberry/Loonies, engendrant de nombreuses productions très inspirantes pour certaines. Les règles étaient simples : afficher le plus de plans possible à 50 Hz dans une résolution minimum. Les plans doivent pouvoir bouger indépendamment les uns des autres (sur les trois axes) et être capables d'avoir chacun leur propre couleur.

En bon demomaker, j'ai donc décidé de me baser sur ces règles pour défier Krusty, et participer à la compétition Oldskool 4K Intro de la Revision 2023 en présentant *Checkmate* qui établit un nouveau record avec 4 plans.

PLAÇONS NOS PIONS

D'apparence simple, cet effet soulève plusieurs problèmes pour le mettre en œuvre sur un CPC : nous ne disposons pas de scrolling hard au pixel ni de bitplanes pour nous faciliter la vie, il va donc falloir ruser.

Un damier est l'illustration parfaite de l'opérateur logique XOR :

XOR	0	1	0	1
0	0	1	0	1
1	1	0	1	0
0	0	1	0	1
1	1	0	1	0

Pour construire un damier, il suffit donc d'appliquer un XOR entre une ligne et une colonne alternant chacune couleur (1) et transparence (0). Notez que celui-ci est composé de 2 lignes différentes qui sont répétées alternativement. Pour plus de commodité, nous appellerons ces lignes « paires » et « impaires ».

Afin de gagner du temps machine, une routine génère au début le l'intro les lignes paires composant un damier pour chaque étape de zoom (les lignes impaires seront obtenues par la suite en y appliquant un XOR). L'effet affichant 128 niveaux de profondeur, ce sont donc 128 lignes de 512 pixels (256 octets) qui sont générées en RAM (dans les banks &c4 et &c5). Il nous suffira donc de choisir la ligne correspondant au niveau de profondeur voulu, et d'en extraire un morceau de 96 octets (largeur de l'écran) en fonction du déplacement horizontal du damier.

Si on s'arrête là, on aura une précision à l'octet pour le déplacement horizontal. On ne va tout de même pas se contenter de ça ! Il va donc falloir sacrifier de la RAM supplémentaire pour copier les 128 lignes générées, décalées d'un pixel, dans les banks &c6 et &c7. À nous les scrollings au pixel !



Les 128 lignes générées en RAM, suivant une perspective 1/2

Dernière précision, les 128 lignes sont réparties en 4 (tiens donc) groupes de 32, chacun ayant une encre différente. En fonction de sa profondeur, un damier ne sera donc pas affiché avec la même encre. Très utile pour la suite !

FOURSOME, AGAIN

Le mode 0 s'est imposé d'emblée en raison de la méthode utilisée pour empiler les damiers. Ça nous arrange bien puisque pour afficher 4 plans, nous aurons besoin de 5 couleurs distinctes (4 pour les damiers et 1 pour le fond).

Le numéro d'une encre en mode 0 est composé de 4 bits (puisque nous en disposons de 16). Ça tombe bien, nous allons pouvoir allouer 1 bit par plan. Vous voyez où je veux en venir ?

Lorsque j'écrivais plus haut que les lignes générées avaient été réparties en 4 groupes, chacun ayant une encre attribuée, celles-ci n'ont pas été choisies au hasard. Si on convertit le numéro de ces encres en binaire, la position du bit mis à 1 correspond à un plan en particulier.

Le plan 1 correspond au plan le plus proche, et le 4^e au plus éloigné :

	Encre
Plan 1	8 %1000
Plan 2	4 %0100
Plan 3	2 %0010
Plan 4	1 %0001

Je rappelle que chaque ligne alterne entre l'encre 0 (transparence) et l'encre qui lui est attribuée. À partir de là, nous allons pouvoir superposer les 4 plans en appliquant un simple OR entre eux !

```
ld a,(hl) ; HL = plan 1
inc h
or (hl)    ; HL = plan 2
inc h
or (hl)    ; HL = plan 3
inc h
or (hl)    ; HL = plan 4
```

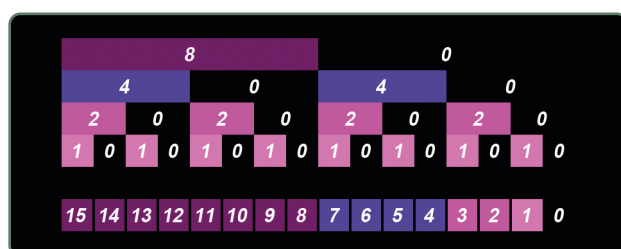
« N'importe quoi, si on empile les 4 plans en suivant cette méthode, on se retrouve avec l'encre 15 (%1111), alors qu'on devrait se retrouver avec l'encre du premier plan ! », s'exclame à juste titre un lecteur tenant à garder l'anonymat. L'astuce réside justement là !

Dès qu'un bit relatif à un plan plus proche que les autres est mis à 1, alors le pixel affiché sera de la couleur de ce plan, peu importe la valeur des autres bits, à condition que les encres correspondant aux différentes combinaisons de ces bits aient la même couleur. Vous me suivez ?

Voici un tableau qui récapitule quelles encres doivent être égales pour chaque plan (avec x = 0 ou 1) :

	Encres		
Plan 1	8 - 15	%1xxx	
Plan 2	4 - 7	%01xx	
Plan 3	2 - 3	%001x	
Plan 4	1	%0001	

Si les encres 8 à 15 ont la même couleur, alors dès que le bit 3 est à 1, on se retrouve avec la couleur du plan 1, peu importe la valeur des bits 0 à 2. Magique non ?



Encres utilisées lorsqu'on superpose 4 damiers à l'aide d'un OR

Les fins connaisseurs y reconnaîtront ici une technique maintes fois utilisée dans les jeux de l'époque pour avoir plusieurs plans à moindre coût (à condition de sacrifier quelques encres).

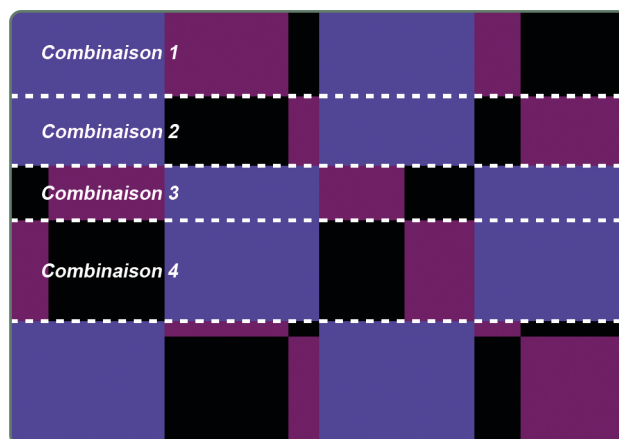
NOMBRE DE SHANNON

À ce stade, nous avons de quoi afficher sur 4 plans une seule ligne qui sera répétée sur toute la hauteur de l'écran grâce à une rupture ligne à ligne, ce qui donne lieu à de bien jolies barres verticales. Cela compose une combinaison de 4 lignes superposées. Or, lorsqu'on observe les 4 damiers, on remarque tout de suite que l'effet est constitué de plusieurs combinaisons de lignes (variant à chaque frame évidemment) !

Il va donc falloir, à chaque frame, générer toutes ces combinaisons à partir de celle de base. Pour cela, nous allons utiliser le fameux XOR dont je vous ai parlé en début d'article.

Imaginons que nous ne souhaitons afficher qu'un seul damier. Celui-ci est composé d'une ligne paire (extraite parmi les 128 lignes stockées), ainsi que d'une ligne impaire obtenue en appliquant un XOR à la ligne paire. Si on ajoute un second plan, composé lui aussi de 2 lignes, la superposition des 2 damiers nécessite alors $2^2 = 4$ combinaisons de lignes.

Pour superposer 4 damiers, nous aurons donc besoin de $2^4 = 16$ combinaisons qui seront astucieusement dispersées dans la VRAM.



Les 4 combinaisons nécessaires pour afficher 2 plans

```

; Combinaison 1, dite de « base »
; Ligne paire plan 1 + ligne paire plan 2

ld d,&00
ld (de),a

; Combinaison 2
; Ligne paire plan 1 + ligne impaire plan 2

ld d,&02
xor %00001100 ; encr 2 (plan 2)
ld (de),a

; Combinaison 3
; Ligne impaire plan 1 + ligne paire plan 2

ld d,&04
xor %11001100 ; encr 1 (plan 1)
ld (de),a ; + encr 2 (plan 2)

; Combinaison 4
; Ligne impaire plan 1 + ligne paire plan 2

ld d,&06
xor %00001100 ; encr 2 (plan 2)
ld (de),a

```

JEU DE PISTE

C'est bien beau d'avoir généré toutes ces combinaisons de lignes superposées, mais comment savoir lesquelles afficher au bon endroit ? Pas de panique, je vais tout vous expliquer.

Comme je l'ai écrit précédemment, les 16 combinaisons générées ont été placées en VRAM à des endroits stratégiques : chaque damier étant composé de 2 lignes (une paire et une impaire), on peut aisément indiquer laquelle on souhaite afficher à l'aide d'un bit qu'on mettrait à 0 ou 1. On peut donc désigner une combinaison en utilisant 4 bits qui indiqueraient chacun le type de ligne (paire/impaire) la constituant.

Ainsi, en utilisant les bits 0, 1, 4 et 5 du registre 12 du CRTC (poids fort de l'offset), nous pouvons placer chaque combinaison au bon endroit en VRAM. La première ligne du tableau (%00000000) correspond à la combinaison de base de 4 lignes :

R12	Offset	VRAM
%00000000	&0000	&0000
%00000001	&0100	&0200
%00000010	&0200	&0400
%00000011	&0300	&0600
%00010000	&1000	&4000
%00010001	&1100	&4200
%00010010	&1200	&4400
%00010011	&1300	&4600
%00100000	&2000	&8000
%00100001	&2100	&8200
%00100010	&2200	&8400
%00100011	&2300	&8600
%00110000	&3000	&c000
%00110001	&3100	&c200
%00110010	&3200	&c400
%00110011	&3300	&c600

Maintenant qu'on sait facilement localiser les combinaisons souhaitées en VRAM, il nous suffit d'actualiser, à chaque frame, une table d'offsets qui sera lue lors de l'affichage de l'effet.

Pour remplir cette table, il nous faut d'abord déterminer la position verticale de chaque damier pour calculer l'offset de départ. Pour chacun d'entre eux, nous allons donc à nouveau utiliser la ligne qui avait été sélectionnée dans la table des 128 lignes et en extraire un second morceau en fonction de la position verticale du damier (comme si c'était une colonne). En vérifiant si le premier pixel de l'extrait horizontal est identique ou pas à celui de l'extrait vertical à l'aide d'un XOR, on peut déterminer la valeur du bit (0, 1, 4 ou 5 selon le numéro du plan) à placer dans l'offset de départ. Lorsqu'on aura traité les 4 damiers, les bits 0, 1, 4 et 5 de l'offset de départ auront été positionnés.

```

; HL pointe sur l'extrait vertical
; NZ = 1 si changement d'encre

52 ** [cp (hl) : call nz,new_ink
      inc l]

; HL' = position dans la table

new_ink
exx
set bit_nb,(hl) ; bit_nb = 0, 1, 4 ou 5
exx
ret

```

En suivant le même principe, nous allons parcourir l'extrait vertical pour chaque damier et à chaque fois qu'un changement d'encre sera détecté (couleur ou transparence), un bit relatif au numéro du plan sera positionné à 1 dans un octet de la table correspondant à la position Y de ce changement.

Lors de l'affichage de l'effet, la table sera lue à chaque ligne pour en déterminer l'offset. Si celui-ci est le même que la ligne précédente, la valeur lue sera 0. Dans le cas contraire, les bits de cette valeur correspondront aux damiers qui alternent entre lignes paires et impaires. Ainsi en appliquant un XOR entre la valeur lue et l'offset précédent, nous obtenons l'offset courant !

```

xor (hl) ; HL = table d'offsets
inc l
out (c),a ; ligne Y
...
xor (hl)
inc l
out (c),a ; ligne Y+1

```

DES DAMIERS ANTHROPOPHOBES

Dernier point à aborder : lorsqu'un damier évolue en suivant l'axe Z, le numéro du plan dans lequel il est affiché change progressivement. Cela se manifeste par un changement d'encre puisqu'on passe d'un groupe de lignes à un autre dans la table de bitmaps générés au départ. Il faut donc penser à faire une rotation de la palette lorsque les damiers changent de plan.

Petit inconvénient lié à cette méthode : 2 damiers ne peuvent pas se retrouver dans la même « zone » sous peine d'un conflit de couleurs, d'autant plus que ça mettrait à mal tout le raisonnement sur les bits vu précédemment. Mais ce n'est pas dramatique.

ZUGZWANG ?

Cet effet ayant été réalisé dans le cadre d'une intro 4K, le player de *Chip'n Sfx* a été utilisé pour jouer la musique, permettant ainsi d'économiser de précieux octets. Inconvénient de taille : ce player n'est pas stable en temps machine et fait de temps en temps d'énormes pics, réduisant ainsi le temps machine pour gérer l'effet. En sortant de la contrainte 4K, on pourrait donc facilement gagner de nombreuses NOPs en utilisant un player stable et plus rapide.

La méthode présentée dans cet article est forcément limitée à 4 plans puisque basée sur les 4 bits d'une encre... Il va falloir trouver une autre technique pour faire mieux ! La VRAM disponible permet de théoriquement afficher 6 plans ($2^6 = 64$ combinaisons), avis aux amateurs de défis ! ■



ELIOT / BENEDICTION

Aujourd'hui, inversons la devise si chère à notre interviewé : « Stop coding, start talking! ». On a pas tous les jours 30 ans... de scène !

PAR HICKS/VANITY

Bonjour Eliot. Tu es un acteur de la scène CPC depuis près de 30 ans, mais on t'entend finalement assez peu t'exprimer. Pourquoi cette discrétion ? À quand un compte TikTok ?

Bonjour, en effet, je passe moins de temps sur le web qu'il y a quelques années, pour plusieurs raisons : il y a une multitude de sites d'actualités, forums, pages à suivre, c'est chronophage, souvent inadapté (Facebook = pas de classement) et au final pas si intéressant. Je trouve dommage que beau-



coup déballent le moindre petit truc fait en 2 heures, sans évoluer vers quelque chose de plus abouti. Il suffit d'ailleurs de constater que les gros projets finalisés, jeux, démos ou utilitaires, ont souvent été réalisés sans déballage public quotidien. Le slogan publicitaire pour les frites McCain « C'est ceux qui en parlent le moins qui en mangent le plus » semble s'appliquer aussi au CPC !

Peux-tu rappeler, en particulier pour les nouveaux venus, ce qui t'a poussé vers le CPC ? C'était en 1993, via le discmag *New Arcade 5* il me semble...

Oui, alors que je bricolais tout seul dans mon coin en BASIC, sur *OCP* et sur *Equinoxe*, je suis entré en contact avec Epsilon pour obtenir le fanzine *New Arcade 4* qui était annoncé dans l'un des derniers *A100%*. Sur la disquette, je lui avais envoyé la seule chose que je pouvais lui envoyer : mes musiques, principalement des adaptations de .MOD Atari ST qu'un ami m'imprimait (oui !!!) et que je recopiais en supprimant une voix. Epsilon a utilisé mes musiques dans *New Arcade 5*, ce qui m'a valu d'être cité dans le dernier numéro d'*A100%* et de rentrer dans Power System puis AsT System. Ces premiers échanges



Digital Press 3 (Benediction)

avec Epsilon mi-1993 ont été déterminants car je n'avais rien vu de la demoscene sur mon CPC avant ! Un monde nouveau m'apparaissait... Le choc de la première démo vue (*Plasma* de Contrast) puis plein d'autres, les fanzines, *The Soundtrækker*, *DAMS*, les lecteurs 3.5 pouces, les meetings, les groupes, le swapping...

C'est vraiment après le premier meeting auquel j'ai participé (Bordelik Meeting 4 en 1995, près de Rennes) que j'ai démarré la programmation en assembleur alors que j'estimais que c'était trop tard pour moi, à 19 ans et sans avoir de cursus informatique, de m'y mettre : un week-end à l'automne 1995 chez Ramlaid avec la présence d'Orphée et du déjà jeune et déjà beau Targhan, qui avaient tous un excellent niveau technique (la *DemolzArt* était déjà sur de bons rails...) et qui m'ont initié aux calculs d'adresses écran, à l'affichage de sprites, aux boucles, aux tests, aux rasters... Preuve de l'importance de tous les meetings et mini-meetings qui sont des moments d'échanges cruciaux ! Une révolution pour moi, donc, ce qui m'a permis en quelques mois avec l'appui de quelques routines d'Epsilon de sortir le discmag *Digital Press 3*.

Quand Benediction est-il né, et à quel moment peut-on considérer que tu as repris le groupe ? Peux-tu nous dire comment vous travaillez ensemble ?

Benediction est né à l'initiative de The Villain (Markus Buntru), un des 2 fondateurs allemands, de *Digital Press*, un discmag dont le premier numéro est sorti fin 1994. À l'époque, les échanges de programmes se faisaient sur disquettes et voie postale mais il était facile d'avoir des contacts CPC à travers toute l'Europe ! The Villain a ainsi eu l'idée de fonder un groupe européen à l'été 1995. Les embauches étaient faciles ! Je me suis donc retrouvé dans ce groupe, alors que je n'avais pas commencé l'assem-

bleur ! Le projet initié par The Villain (qui ne codait pas) avec l'assistance technique de The Woodman (le codeur de *Digital Press*) était une mégadémo appelée *European Megademo*. Pour mettre en application mes premières lignes de code Z80, j'ai produit fin 1995 l'invitation officielle qui est aussi la première production estampillée Benediction. Je ne sais pas comment l'invitation a été diffusée et quelles sont les personnes qui ont réellement bossé sur une part (pour sûr, Shap/Overlanders et le groupe D-Zign) mais The Villain et The Woodman ont quelque peu disparu de la circulation. Sans moteur, l'*European MD* a rejoint la liste des mégademos inachevées (*Take Off* de Beng!, *Cuddly*, ...). Seul Antitec (Grèce) produisait à cette période sous le nom du groupe. Je me suis donc retrouvé plutôt es-soulé, après *Digital Press 3* (mars 1997) qui représente ce que Benediction devait être : des compétences complémentaires et de pays divers (Rainbird, Greg, The Villain, Antitec, Eliot, ...). J'ai donc évolué seul, une idée d'effet allant souvent jusqu'au bout et sortant dans une démo en collaboration avec les gens actifs du moment (*System Party*, *Simply The Bests!*, *20-21-22*, *What A Cool Week-End!*, ...).

L'arrivée de Krusty (Romain) qui débutait a été stimulante, car on habitait pas loin à l'époque. Il a rapidement progressé et, avec un regard nouveau, s'est attaqué à des effets orientés mathématiques dès 2002 avec la démo 23-24-25 qui marque la première démo Benediction à deux codeurs ! On a continué assez régulièrement à sortir des démos, toujours principalement pour les meetings, mais sans coordination réelle entre nous. C'est lors d'un mini-meeting chez moi en 2010 avec Targhan et Supersly qu'une idée a germé chez Krusty et a donné naissance à un effet de la future *Wake Up!*, le rotozoom je crois. Krusty était devenu très expérimenté en cross development et en outils modernes, cela permettait d'envisager une grosse démo sans chargement avec de nombreux effets compactés en mémoire.



Simply The Bests! (Benediction)

N'ayant pas la même maîtrise des outils modernes (parfois incompatibles avec ma machine de développement), il y a eu beaucoup de bricolage de mon côté car les sources étaient assemblés avec *Winape* chez moi et avec *Sjasmpius* chez Krusty, créant parfois des bugs qui n'avaient pas lieu d'être... Moi qui aime maîtriser tout ce qui se passe dans mon code, c'est un peu perturbant ! Mais on s'en est quand même bien sorti, malgré ça et ma disponibilité très aléatoire ! L'arrivée chez Benediction d'Exin (depuis *Oh No! 2*) et de Voxfreax (depuis *Plasmorphy*) a permis d'avoir un contenu graphique original et un design lumineux (aucun fond noir). Pour notre part pour la *30 Years MD*, *Quest For Perfection*, on a fonctionné avec des outils compatibles entre nous (*Git*, *Make*, pour les connaisseurs) : je voyais réellement ce que je codais ! Une révolution ! :) Krusty a codé le scrolling text et moi tout le reste. Là encore, ma disponibilité m'a joué des tours mais c'est une démo riche en configurations CRTC et qui m'a énormément fait progresser. Cette démo a permis à Voxfreax d'aller plus loin que le simple dessin de logo ou le choix de palettes : il y a une grosse fonte (raccourcie par la suite de 3 ou 4 pixels pour passer le scroll sur 8 Ko de vidéo au lieu de 16...) ainsi que des motifs pour magic rasters.

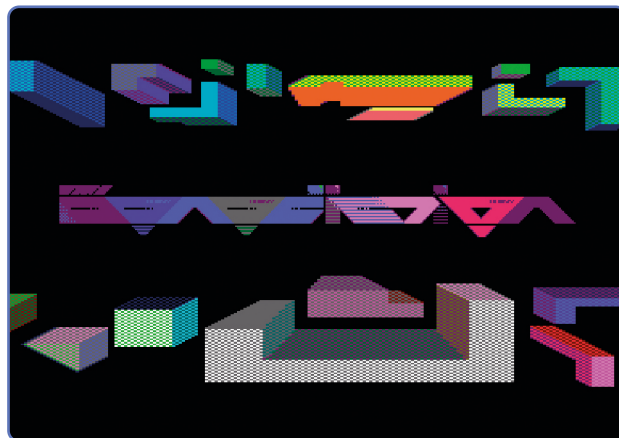


Quest For Perfection (Benediction)

D'un point de vue de fonctionnement, vous l'avez compris, Krusty a la maîtrise des usines à gaz. Avec les outils modernes, il peut modifier une ligne d'une source et en 1 clic de souris, assembler, compacter les données et générer un .DSK ou un .SNA de la démo et la lancer sur son CPC pour un test immédiat... Ça permet d'imaginer des choses puissantes et lourdes que beaucoup n'oseraient pas tenter en développement natif. De mon côté, je me sens plus artisanal, plus dans le détail, prêt à lancer 12 fois OCP pour déplacer en douce quelques pixels ou inverser localement 2 couleurs sur un dessin de Voxfreax pour caler correctement des split rasters. Pour l'instant, je trouve qu'on n'a pas encore eu l'occasion de mettre vraiment en pratique cette complé-

mentarité sur une production en particulier mais c'est un truc à travailler !

Votre dernière démo d'envergure, *Can Robots Take Control?*, a impliqué de nombreuses années de développement. Êtes-vous satisfaits du résultat, et de la réception par la scène ? Avec le recul, y a-t-il des choses que vous feriez différemment ?



Can Robots Take Control? (Benediction)

Oui, cette démo est un gros morceau dont le mérite revient principalement à Krusty et Voxfreax car, à cette période, je n'ai pu travailler que sur 2 écrans, l'intro qui ne devait être qu'un écran fixe au départ et le soldat en mode 1 split rasterisé. Krusty a montré des effets puissants, inédits. Voxfreax a réalisé un sacré paquet de graphismes, de travail sur les palettes et le script. Et la cerise sur le gâteau est la musique du toujours jeune et toujours beau Targhan réalisée à la toute fin en un week-end ! L'accueil a été bon et au-delà du CPC, ce qui prouve que le CPC est aussi regardé par les autres scènes ! Il faut du temps pour digérer une prod et j'espère qu'on verra d'autres chaos zoomers ou tunnels ! Sur le principe, on peut toujours faire mieux mais il y a tellement de séquences différentes que ce serait interminable si on voulait que tout soit au même niveau technique ou esthétique. C'est le piège de ce genre de grosses productions et on peut ressentir la même chose pour *Batman Forever* ou *phX*. Au niveau visuel, j'ai échangé *a posteriori* avec Voxfreax sur le choix de fond noir qui donnait peut-être un aspect vide. Bref, on échange et on s'améliorera ! La question à se poser est celle du format. Personnellement je préfère un format à la *Wake Up!*.

Tu as une longue expérience de la scène CPC. Quel est ton avis sur l'évolution des démos ? Quelle est ta conception d'une démo réussie ?

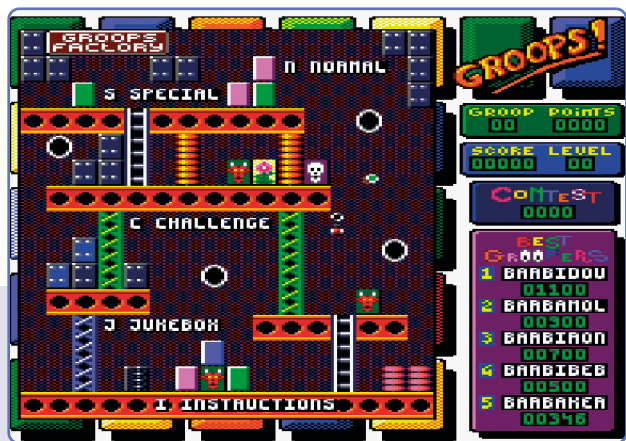
Les démos CPC sont moins nombreuses mais globalement elles progressent techniquement et esthé-

tiquement. Des records sont battus, l'esprit demomaking est toujours là ! Le CRTC, entre autres, est mieux maîtrisé mais parfois sans inspiration probablement parce que le couple codeur/graphiste existe trop rarement dans les projets CPC alors que ça changerait tout ! Imaginons *X-Kore* avec un designer... Le visuel, le son et le FDC doivent aussi battre des records ! De nouveaux concepts sont encore à trouver, il faut tester, travailler dur... Sur ce sujet, les travaux de Supersly, Beb, BSC ou roudoudou sont des avancées remarquables et inspirantes. De plus, il se dit aussi dans les milieux autorisés que le mélange hard + soft n'a révélé pour l'instant que 6 % de son potentiel, ce qui laisse de la marge de progression pour les futures démos et jeux...

Pour moi, la clé d'une démo réussie pour le spectateur est « Woowow, comment ils font ça ? ». J'apprécie les choses malines qui font la différence avec la technique brute. Je pense par exemple aux animations dans *Logon's run*, à l'utilisation du border dans la part 2 de *Onescreen Colonies #1*, aux différentes astuces de Madram (voir *Balade dans le monde de la démo* dans le numéro 1)... Bref, sans astuce maline, l'effet est brut, technique et la démo moins spectaculaire. Je suis persuadé que tout demomaker CPC, même retiré, est sensible à ça et que ça doit forcément le titiller quand il voit un truc « malin » !

Parlons jeux. Tu en as codé plusieurs (*Solomon's Key 3*, *Push*, *Groops!*, *Sudoku Master*), mais plus aucun depuis presque 15 ans. Alors, à cours d'idées, de graphiste, ou de temps ? Que penses-tu par ailleurs des jeux sortis ces dernières années ?

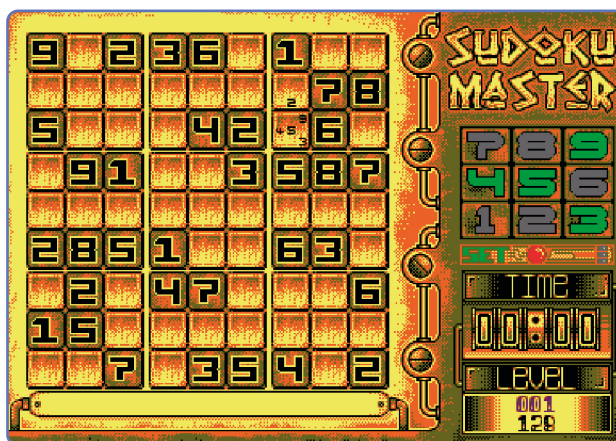
J'ai codé des jeux alors que peu de personnes produisaient des jeux. *Groops!* et *Sudoku Master* ont une approche « démo ». J'ai voulu encourager le fullscreen comme standard pour des jeux de plateau, intégrer des changements de mode, split rasters, digidrums... J'ai eu la chance de travailler avec Slyder (*Groops!*), CeD (*Sudoku Master*) et



Groops! (Benediction)

Napo (les deux jeux), qui ont été besogneux et d'une efficacité remarquable. Les animations de Slyder pour *Groops!* sont par exemple le fruit d'un gros travail, de nombreuses retouches jusqu'à la perfection. Depuis je n'ai pas eu d'idée, et encore moins de temps, pour de nouveaux jeux.

Personnellement, je ne suis pas du tout joueur, sauf de jeux de réflexion, et pour un jeu plus complexe, il me faudrait quelqu'un pour gérer le gameplay, car je ne suis pas sensible à la chose ! Ma priorité reste la démo. Certains opposent le jeu et la démo arguant qu'un jeu est plus difficile à faire qu'une démo ou qu'un jeu n'a pas besoin de l'esbrouffe technique d'une démo. C'est surtout que faire un jeu est plus long que faire une démo et on voit de gros projets de jeux s'étaler sur des années : j'avoue, je ne pourrais pas ! Pourquoi pas plutôt un *Fruity Frank* en 4 Ko ? ;)



Sudoku Master (Benediction)

Concernant la pléthore de jeux sortis ces dernières années, il faut admettre que la réalisation est de plus en plus correcte, l'expérience est là, ça fonctionne, sans bug, c'est propre mais on ne fait toujours pas mieux que *Prehistorik 2* sorti en 1993. Des techniques issues de la démo ne sont pas encore intégrées aux jeux. Pourquoi pas des split rasters, des scrollings hard différentiels, de la rupture ligne à ligne ? Pour revenir au jeu en lui-même, j'aime beaucoup les derniers shoot'em up (*Alcon 2020* et *Red Sunset*), *Baba's Palace*, *Imperial Mahjong*, *Be Tiled!* qui sont efficaces à tout âge et on se laisse prendre à pas mal de jeux type *Gate 2 Heaven 2*. Les Espagnols ont quand même un sacré coup de patte sur tous les plans (il y a des leçons à prendre !) même s'il manque à mon goût une profondeur sur tous ces jeux de plateformes qui se ressemblent trop. Je vieillis ?

La production pour le concours CPC Retro Dev est assez bluffante aussi. Réalisés par des « amateurs », certains concepts originaux mériteraient une meilleure finition.

Quels sont les projets en cours au niveau personnel et plus collectivement chez Benediction ?

J'ai plusieurs projets en cours, j'espère qu'un premier onscreen sera sorti avant la publication de ce numéro (NDLR : *Stand Up!*) puis une démo un peu plus conséquente sur laquelle il me reste pas mal de finitions. J'ai des effets qui sont gardés au frais pour une prochaine démo Benediction. Je sais que je vais avoir plus de temps libre pour coder et idéalement intégrer la toolchain de Krusty...

Tu as organisé plus de 20 meetings en un peu plus de 20 ans. Quelle est l'importance de ces événements, pour toi et pour la scène ? Constates-tu une évolution au niveau du public ou de ce qui s'y passe ?

Quand je suis arrivé, il y avait de nombreux rassemblements CPC, j'ai participé à un grand nombre d'entre eux, c'était vital pour apprendre, échanger, faire naître ou avancer des projets. J'ai rapidement organisé des « mini-meetings » chez moi, puis un gros meeting Camembert 4 en 1998. Contrairement à ce que beaucoup pensent, ce n'est pas compliqué d'organiser un meeting. Il faut juste trouver de bonne heure la salle ! Avec les Amstrad Expo, le ciblage était « grand public » car en 2000, il y avait encore des utilisateurs solitaires nostalgiques d'Amstrad ou tout du moins le nom attirait encore. Au fil du temps, ça s'est étioilé, les meetings redevenant des événements de passionnés actifs de la scène. Avec la Benediction Coding Party #1, ce fut plus cosy, petit gîte pour cause de Covid, mais assez productif. J'encourage d'autres personnes à organiser ce genre de rassemblements, petits ou grands ! Un

Benediction Coding Party #1



grand appart, une grande maison ou un gîte suffisent... Ça vaut des milliers d'heures de chat sur Internet !

Suis-tu la scène démo en dehors du CPC, et si oui quelle leçon devrions-nous tirer d'elle ? Une autre machine de prédilection ?



Benediction Coding Party #1

Je possède plusieurs autres machines : Atari STe, 800XL, Spectrum 128 et C64 mais je regarde principalement ce qui se fait sur C64. C'est vraiment inspirant en terme d'idées d'effets inédits à transposer sur CPC. Une chose qui me plaît particulièrement est qu'il n'est pas rare de revoir un effet déjà sorti mais amélioré par le même groupe ou un autre, ce qui est un des ressorts principaux du demomaking (le challenge !), un autre étant la créativité, dont ils font également preuve...

Merci pour tes réponses ! Je te laisse un dernier espace d'expression libre...

Merci pour ces questions, ce fut l'occasion de ressortir pas mal de souvenirs agréables et d'exprimer le fait que, depuis mon arrivée, j'ai la chance de côtoyer, souvent trop furtivement, un paquet de gens impressionnants par leur niveau et ce qu'ils apportent au CPC en créant ou en nous permettant de le faire. Merci donc à tous les concepteurs d'extensions (X-Mem, Albireo, HxC, etc.), de logiciels adaptés (*Orgams*, *Arkos Tracker*, le *Soundtracker DMA*, *IMPDraw*, *CPCConvImg*, etc.), d'UniDOS, d'émulateurs, d'assembleurs et à ceux qui oeuvrent pour préserver (CPC-Power, ACME, etc.) et partager leurs connaissances ! ■



LOADER BASIC BINAIRE

Quiconque a déjà généré un exécutable binaire s'est déjà posé la question de la création d'un loader... et aura inévitablement pesté contre un « MEMORY FULL » ou un « Drive A : disc missing ».

PAR OFFSET/FUTURS'

UN PROBLÈME TRIVIAL ?

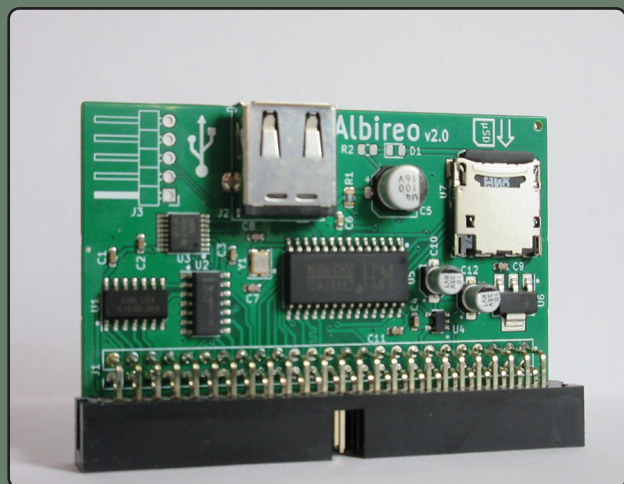
« Mouahaha ! Trop facile ! » On dirait bien que le petit blond à lunettes traîne encore dans les parages... Eh bien non ! Ça n'est pas si facile ! Et il y a de nombreux pièges, moult façons de mal faire et de se faire mal !

Le premier réflexe est de simplement lancer le binaire via un `RUN"MON-BIN.BIN"`. En effet, le

BASIC prend parfaitement en charge l'exécution des programmes binaires, et ça va fonctionner... dans les cas les plus élémentaires uniquement. Car on va vite se heurter à des limites – très vite même.

Le premier souci, c'est que lorsqu'il exécute un binaire, le BASIC fait un `MC BOOT PROGRAM`. Ce mode de lancement permet de s'assurer que la main est donnée au binaire dans le cadre d'un système propre, comme si la machine venait de démarrer, avant même le lancement du BASIC lui-même. Or donc, pas de `MEMORY` ou de `SYMBOL AFTER` qui traîne, pas de couleurs alambiquées, et surtout, pas de ROM initialisées !

« Ouais ? Et alors ? Il est où le problème. » Décidément, toujours aussi agréable ce petit blond. Eh bien le problème est justement là, au niveau des ROM. Pas de ROM, ça signifie également : pas d'AMSDOS. Si votre binaire est un programme indépendant qui n'a rien besoin de charger, ça sera sans effet, et vous pourrez vous contenter joyeusement de cette méthode. Mais si vous avez besoin d'accéder aux fichiers, point de salut, il va falloir réactiver l'AMSDOS. Et c'est là que les ennuis commencent.



L'Albireo

AMSDOS, où ES-TU ?

L'AMSDOS se trouve historiquement en ROM numéro 7. Il est donc coutume, dans votre binaire, d'initialiser spécifiquement cette ROM via un appel à KL INIT BACK.

```
INIT_ROM_7
ld c,7      ; C = ROM AMSDOS par défaut
ld de,&40    ; DE = LOMEM par défaut
ld hl,&abff  ; HL = HIMEM par défaut
call &bcce  ; KL INIT BACK
ret
```

Ce code pose toutefois deux problèmes.

Le premier, bien connu de tous les utilisateurs de lecteurs B, fait que le lecteur courant est lui aussi réinitialisé. Qui n'a jamais pesté devant une démo ou un jeu réclamant la disquette dans le lecteur 3" en A alors que vous l'aviez lancé depuis le lecteur 3"1/2 en B ? Heureusement, il existe une solution simple. Comme le MC BOOT PROGRAM ne touche pas à la mémoire, celle-ci contient encore l'information sur le lecteur courant. On peut donc le récupérer afin de pouvoir le restaurer une fois l'AMSDOS réinitialisé.

Notre routine devient alors :

```
INIT_ROM_7
ld hl,&be7d  ; HL = Adresse de la mémoire de travail
              ; de l'AMSDOS
ld a,(hl)    ; A = lecteur courant (typiquement 0 ou 1)
push af
ld c,7       ; C = Numéro de ROM AMSDOS par défaut
ld de,&40     ; DE = LOMEM par défaut
ld hl,&abff   ; HL = HIMEM par défaut
call &bcce   ; KL INIT BACK
pop af
ld hl,&be7d  ; HL = Adresse de la mémoire de travail
              ; de l'AMSDOS
ld (hl),a    ; Sélection du lecteur courant
ret
```

« Eh ben voilà ! C'était pas si compliqué ! » Non mais quel nabot ce petit blond à lunettes ! On a dit qu'il y avait *deux* problèmes ! Et le second est autrement plus ennuyeux.

Si l'AMSDOS est placé en ROM 7 par défaut, il est des cas où le système a été étendu avec d'autres ROM permettant de gérer d'autres supports comme l'Albireo, la X-Mass ou la M4 Board ; ou même simplement d'autres formats de disquettes via ParaDOS. Et si les utilisateurs exigeants auront remplacé l'AMSDOS par une ROM de gestion compatible à l'emplacement 7 (UniDOS ou ParaDOS par exemple), comme l'opération n'est pas triviale, ça ne sera pas toujours le cas.

En résumé, si vous aviez lancé votre binaire depuis votre clé USB branchée sur votre Albireo, paf ! Une fois l'AMSDOS réinitialisé, adieu la gestion des nouveaux supports ! Et voilà qu'à nouveau vous est réclamée une disquette dans le lecteur A !

Qu'à cela ne tienne ! Nous allons donc réinitialiser *toutes* les ROM !

```
INIT_ALL_ROM
ld hl,&be7d
ld a,(hl)
push af
ld de,&40    ; DE = LOMEM par défaut
ld hl,&abff  ; HL = HIMEM par défaut
call &bccb   ; KL ROM WALK
pop af
ld hl,&be7d
ld (hl),a
ret
```

Le petit blond à lunettes se frotte les mains. Oui, effectivement, cette solution va fonctionner la plupart du temps. Sauf qu'il peut y avoir beaucoup de ROM installées sur un système, et que celles-ci consomment de la mémoire, affichent des messages à l'écran, etc. De plus, si la ROM qui gère

notre disque revient à son état par défaut (comme le fait l'AMSDOS), on ne pourra pas davantage continuer nos chargements depuis le disque ou lecteur courant.

Bref, c'est tout sauf robuste, tout sauf élégant.

LOADER BASIC, NIVEAU PRIMAIRE

Demeure la solution de faire un bête loader BASIC. Le même que celui que le petit blond à lunettes est en train de taper béatement en ce moment :

```
10 MEMORY &7FFF
20 LOAD"MON.BIN",&8000
30 CALL &8000
```

Dans ce cas précis, votre binaire est lancé dans l'environnement courant et bénéficie immédiatement de vos ROM actives (quelles qu'elles soient). Pas besoin de s'embêter à réinitialiser l'AMSDOS, ni même à se soucier si c'est lui qui est en charge du disque ; tout roule tout seul. Mais là encore, malgré une apparente simplicité, c'est un nœud à problèmes.

Le premier est purement fonctionnel : impossible de charger un binaire qui descendrait trop bas en mé-

moire (vous seriez immédiatement puni par un « MEMORY FULL ») ; impossible donc d'avoir un gros binaire de plus de 40 Kio. Le second problème est du point de vue des performances et du stockage : on charge deux fichiers au lieu d'un, le loader et le binaire, et bien sûr, on a deux fichiers sur le disque.

Plus lent, plus gros. Bref, la misère.

LOADER BASIC, NIVEAU COLLÈGE

Ne comptez pas sur le petit blond à lunettes pour ça, mais, au moins pour le premier problème, il est possible d'améliorer les choses en truandant gentiment le système.

```
10 SYMBOL AFTER 256
20 OPENOUT"dummy"
30 MEMORY &1ff
40 LOAD"MON.BIN",&200
50 CALL &200
```

On a là deux bidouilles bien connues. D'abord, on fait remonter le HIMEM de quelques dizaines d'octets grâce à un SYMBOL AFTER 256 (la valeur par défaut est 240, ce qui réserve l'emplacement mémoire pour 16 caractères reconfigurables, inutiles pour nous).

Ensuite, de manière assez subtile, on permet le chargement des binaires beaucoup plus bas en mémoire à l'aide d'un OPENOUT"dummy" bien placé. Celui-ci force l'allocation du tampon mémoire qui sert pour entrées/sorties ; 4 Kio qui se retrouvent réservés juste au-dessous du HIMEM avant qu'il ne soit rabaisé par le MEMORY. Sans cette astuce, cette mémoire est allouée sous le MEMORY ; ce qui provoque un « MEMORY FULL » incongru, bien connu de notre petit blond incompetent. À noter que ce tampon n'est même pas utilisé lors du chargement d'un fichier binaire ! En outre, vous constaterez que ce fameux OPENOUT"dummy" ne provoque aucune écriture disque vu que ledit fichier n'est jamais fermé.

C'est bien beau tout ça, mais demeure le problème des deux fichiers ! Voire *trois* maintenant ! Car même s'il n'est pas créé, l'ouverture du fichier « dummy » génère un accès disque, et donc de la lenteur.

Vous en conviendrez, ça n'est pas joli joli !

LE BINAIRE MASQUÉ

La solution ultime est donc d'embarquer votre binaire *dans* le loader BASIC ! Vous bénéficiez alors de l'avantage d'un loader BASIC (pas de ROM à ré-initialiser, environnement utilisateur par défaut) et du

lancement direct du binaire (possibilité d'un binaire gros et/ou en mémoire basse, un seul fichier à lancer, un seul fichier sur le disque).

Et cerise sur le gâteau, on peut en profiter pour faire un loader décoratif dans lequel on masquera toutes les parties techniques, ne laissant apparaître que des messages. Avec ce petit côté magique qui explique sans doute le filet de bave sortant de la bouche du petit blond à lunettes.

Imaginez un programme qui s'affiche ainsi :

```
LOAD"BAS-BIN"
LIST
10 ' Exemple de BASIC/Binaire
20 ' @2023 64 NOPs
```

Mais qui s'exécute comme ça :

```
RUN"BAS-BIN"
On est dans l'assembleur !
On est de retour au BASIC !
```

Sympathique non ?

Eh bien pour ce faire, rien de plus simple ! Nous allons directement générer le programme BASIC en assembleur, y intégrer un programme purement binaire, et masquer les lignes que l'on ne veut pas voir au LIST !

Voici notre programme BASIC effectif (où &xxxx sera une adresse générée automatiquement vers notre binaire intégré à la fin, où les lignes grisées seront masquées) :

```
10 ' Exemple de BASIC/Binaire
20 ' @2023 64 NOPs
30 ' CALL &xxxx
40 PRINT"On est de retour au BASIC !"
```

Chaque ligne de programme BASIC est composée comme suit :

- 1 mot avec la taille de la ligne en octets.
- 1 mot avec le numéro de la ligne.
- x octets avec le contenu de la ligne tokenisé.
- 1 octet de fin de ligne (valeur &00).

Et à la toute fin du programme, deux octets à 0 qui en indiquent la fin.

À noter que l'on ne va pas détailler ici la façon dont le BASIC encode ses lignes avec les tokens. L'exemple ci-après n'en utilise de toute façon que quelques-uns qui sont relativement simples à appréhender.

Pour le reste, il y a quand même deux petites choses à savoir pour bien comprendre le principe général. La première, c'est que le BASIC va charger la totalité de votre fichier .BAS, y compris ce qui se trouve après la fin du programme. Il est donc tout à fait possible d'y placer du code assembleur ou des données quelconques. Après un LOAD ou RUN, ceux-ci se retrouveront tout naturellement en mémoire à la suite du programme BASIC.

La seconde, c'est que lors du RUN, le BASIC va exécuter toutes les lignes de votre programme, jusqu'au marqueur de fin, sans tenir compte de la taille annoncée. Alors que lors d'un LIST, le BASIC va sauter de ligne en ligne selon la taille annoncée pour les afficher ! De ce fait, si vous annoncez une taille de ligne qui englobe les lignes suivantes, cela n'aura absolument aucun impact sur le RUN, mais aura pour effet de rendre lesdites lignes invisibles au LIST !

Et donc, voici un petit programme d'exemple qui devrait fonctionner avec la majorité des assembleurs actuellement sur le marché !

Pas de difficulté particulière, sinon que le programme est assemblé en &8170 comme s'il était en &170 (l'adresse de base des programmes BASIC), puis sauvé en tant que programme BASIC via la routine en &8000. En résumé, vous assemblez, puis vous faites un CALL &8000, et voilà ! Un fichier BAS-BIN.BAS sera sauvé qui contient votre BASIC avec lignes masquées et binaire intégré !

Vous n'avez désormais plus aucune excuse pour ne pas créer de beaux loaders, compatibles avec tous les lecteurs, toutes les ROM de gestion disque, et joliment décorés ! Je vois d'ailleurs que le petit blond à lunettes est déjà au travail. Vous aussi ? ■

```
; Exemple de génération d'un programme BASIC
; avec un binaire intégré et des lignes masquées
;
; Par OffseT/Futurs' pour 64 NOPs (09/2023)
```

```
org &170,&8170
```

```
; Quelques vecteurs système utiles
```

```
TXT_OUTPUT      Equ &bb5a
CAS_OUT_OPEN     Equ &bc8c
CAS_OUT_DIRECT   Equ &bc98
CAS_OUT_CLOSE    Equ &bc8f
```

```
; Quelques tokens de codage du BASIC
```

```
TOKEN_FinDeLigne Equ &00
TOKEN_Separateur  Equ &01
TOKEN_NombreHexa16Bits Equ &1c
TOKEN_CALL        Equ &83
TOKEN_PRINT       Equ &bf
TOKEN_Apostrophe  Equ &c0
```

```
DebutBasBin
```

```
; 10 ' Exemple de BASIC/Binaire
```

```
Ligne10 dw Fin10-$           ; Taille de la ligne
dw 10                        ; Numéro de ligne
db TOKEN_Separateur          ; Contenu de la ligne
db TOKEN_Apostrophe
db " Exemple de BASIC/Binaire"
db TOKEN_FinDeLigne          ; Marque de fin
```

```
Fin10
```

```
; 20 ' @2023 64 NOPs
```

```
Ligne20 dw FinProgramme-$    ; Ici on altère la taille
dw 20                        ; pour masquer toutes les
db TOKEN_Separateur          ; lignes qui suivent la
db TOKEN_Apostrophe          ; ligne 20 lors du LIST
db " ",&a4,"2023 64 NOPs" ; du BASIC
db TOKEN_FinDeLigne
```

```
Fin20
```

```
; 30 ' CALL &xxxx
```

```
Ligne30 dw Fin30-$           ; Ligne formée normalement
dw 30                        ; mais qui sera masquée
db TOKEN_CALL,TOKEN_NombreHexa16Bits
dw CodeAssembleur
db TOKEN_FinDeLigne
```

```
Fin30
```

```

; 40 PRINT"On est de retour au BASIC !"

Ligne40 dw Fin40-$
        dw 40
        db TOKEN_PRINT,34,"On est de retour au BASIC !",34
        db TOKEN_FinDeLigne
Fin40

FinProgramme
        dw 0 ; Marqueur de fin de programme

; Code assembleur intégré à la fin du BASIC

; Une simple routine qui affiche un texte

CodeAssembleur
        ld hl,TexteAssembleur
Boucle  ld a,(hl)
        or a
        ret z
        call TXT_OUTPUT
        inc hl
        jr Boucle

TexteAssembleur
        db "On est dans l'assembleur !",10,13,0

FinBasBin

; Fin effective de notre fichier BASIC avec binaire intégré

; Routine de sauvegarde du programme
; BASIC avec le binaire intégré qui
; a été assemblé en &8170

Org &8000

SauveBasicBinaire
        ld hl,AdresseNomFichier
        ld b,LongueurNomFichier
        call CAS_OUT_OPEN ; Création du fichier BASIC

        ld a,0 ; Fichier de type BASIC (0)
        ld hl,&8170 ; Adresse où se trouve le BASIC
        ld de,FinBasBin-DebutBasBin ; Longueur du fichier
        ld bc,0 ; Pas d'adresse d'exécution
        call CAS_OUT_DIRECT ; Écriture du fichier BASIC

        call CAS_OUT_CLOSE ; Fermeture du fichier BASIC
        ret

LongueurNomFichier Equ 11
AdresseNomFichier db "BAS-BIN.BAS"

```

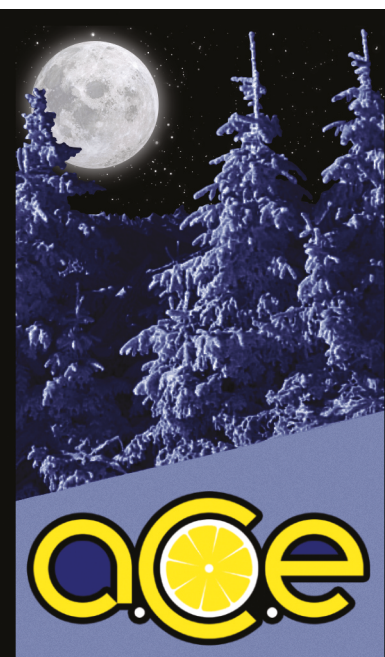


LE JOUR, ET...

Le meilleur Assembleur.

- Compresseurs intégrés, compression de code à la volée.
- Formats CPC natifs gérés (cartouche, snap, disk, cassette, flottants 40 bits).
- Vitesse légendaire.
- Exports spéciaux pour ACE.

Dispo sur      



LA NUIT.

Le meilleur Emulateur CPC/ +.

- Nombreux outils et fonctionnalités.
- Intégration avec RASM pour un débog précis et intuitif.
- Vitesse légendaire.
- Des plug-ins sont à venir mais le meilleur est déjà là...

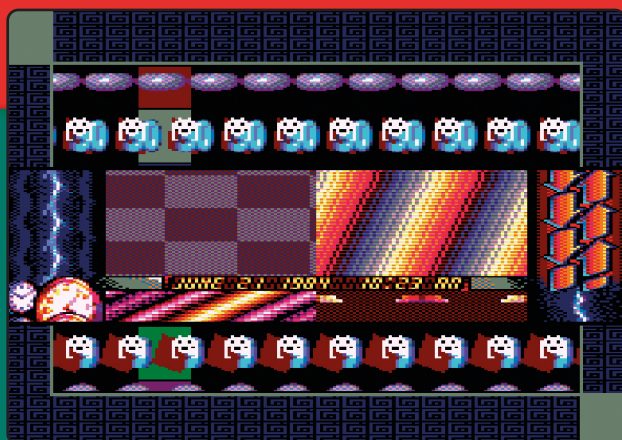
Dispo sur      



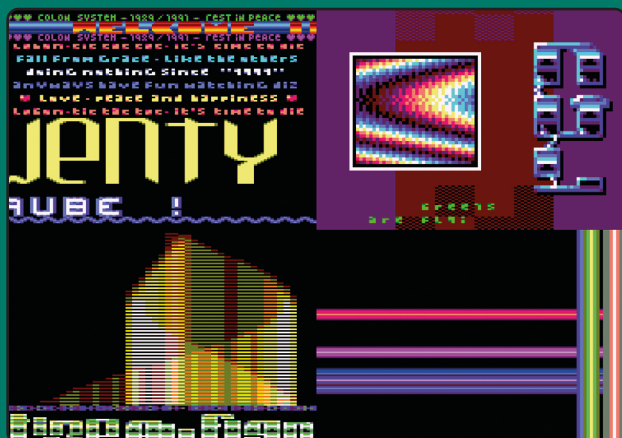
VIVE LE « CRTC 1 ONLY » !

Fidèle lecteur de *Demoniak* dans mes jeunes années, la rubrique « Humeur » était l'une de mes préférées : on y voyait Targhan et Orphée polémiquer joyeusement sur un sujet d'actualité. Avec la même logique, venons aujourd'hui au secours de notre pauvre CRTC 1 et clarifions quelques points à propos de la programmation de notre contrôleur de tube cathodique favori.

PAR HICKS/VANITY



Back To Futurs' (Futurs')



Chapelle Sixteen (Vanity)

Quelques CPCistes s'indignent périodiquement que certaines démos ne soient compatibles que sur CRTC 1.

— Des noms, des noms !

— Oui : *From Scratch* (Vanity), *Chapelle Sixteen* (Vanity), *Back To Futurs'* (Futurs'), et plus récemment *The One* (Impact) et la seconde partie de *Onescreen Colonies #3* (Vanity, encore ?!)¹.

— Mais non, on veut les noms des CPCistes indignés, t'es con ou quoi ?!

Quoi qu'il en soit, de fines explications abondent : « Leurs auteurs sont trop mauvais pour les adapter ! », « Ils recherchent la facilité, ces fainéants ! », « Ce ne sont pas de vraies démos ! », « Ça ne fonctionne que sur un CPC sur cinq », ou « Sur CRTC 1, tout est plus simple ! ».

¹ On pourrait ajouter à cette liste *DSC #4* (Logon System), puisque son auteur indique lui-même, après avoir fustigé le « CRTC 1 only » pendant des années, qu'il faut la lancer sur CRTC 1, sans quoi « l'effet visuel est simplement ennuyeux à regarder » (source : Pouet).

PLUSIEURS CRTC ?

Calme-toi René, je vais tout t'expliquer. Le CPC possède plusieurs versions de CRTC, le chipset s'occupant de gérer la structure de l'écran et ses signaux de synchronisation vidéo, qui ont été numérotées de 0 à 4. Soit 5 variantes principales, avec des changements plus ou moins importants. Parmi eux, le CRTC 1 possède plusieurs particularités :

- La prise en compte de R4 et R9 s'effectue sans délai. Ainsi, mettre R9 = 0 quand C9 = 0 donne bien C9 = 0 à la ligne suivante,
- L'offset est pris en compte quand C0 = C4 = 0 (et non quand C0 = C4 = C9 = 0 comme sur CRTC 0) ou quand C0 = 0 si R1 > R0. Cela permet de le modifier sans ligne à ligne à certaines conditions,
- Le border peut être activé avec R6, mais il est impossible de l'activer puis de le désactiver quand C4 = 0, contrairement au CRTC 0 (qui nécessite R8).

On ne voit alors pas pourquoi la programmation CRTC devrait consister à chercher la compatibilité avec toutes les versions, et non l'exploration des forces et des faiblesses d'un CRTC particulier. René est calmé, on peut commencer l'analyse de détail.

POURQUOI SONT-ILS AUSSI MAUVAIS ?

Back To Futurs' et *The One* contiennent de la RVMB : or, sur CRTC 0, le dernier caractère de chaque écran est entaché d'un bug de border, et la juxtaposition de ces écrans n'est plus parfaite. Avec beaucoup d'efforts, *The One* aurait pu fonctionner, c'est-à-dire tourner sans planter, mais serait pleine d'artefacts. C'est le cas de *(out)dated* (Semilanceata) lancée sur un CRTC 0, que certains ont tout de même clouée au pilori pour cette raison, malgré les efforts de Grim pour l'adapter au mieux. Pour *Back To Futurs'* c'est pire : elle tire profit du fait que l'offset est pris en compte quand C0 = C4 = 0, quel que soit C9, ce qui la rend inadaptable sur un autre CRTC que le 1, puisque c'est le seul à présenter cette caractéristique. Dans ce cas, c'était la démo CRTC 1 only, ou pas de démo du tout.

Chapelle Sixteen est un abracadabrantesque concentré de ruptures diverses (classique, ligne à ligne, verticale) : c'est seulement à ce prix que la simulation de 4 écrans indépendants est possible. Pour la rendre compatible CRTC 0, il aurait fallu recoder toute la démo, tellement les timings sont serrés, et les accès CRTC nombreux. De plus, des bugs à la jonction des écrans seraient apparus un peu partout, cassant totalement le design. Comme précédemment, on réclame un effort déraisonnable pour un résultat dégradé.

Halte aux fake news ! Il n'existe pas jusqu'à présent de façon simple de faire disparaître l'artefact de border situé à la fin d'une ligne sur CRTC 0, et qui se révèle surtout gênant quand on exploite la rupture verticale. Une solution, découverte il y a bien des années par Madram, et récemment reprise dans le *Compendium* de Longshot, implique plusieurs OUT par écran (sur la ligne), ce qui est totalement incompatible avec la programmation d'une démo aux timings serrés, sauf à concevoir un effet très basique. L'idée est théoriquement intéressante, mais pratiquement inexploitable : s'il y a jonction, il y a rupture verticale, donc manipulation de nombreux registres en 64 NOPs (en général R12 et/ou R13, et R2 ou R3, voire R0). Certains arguent qu'il vaut mieux une démo moins impressionnante compatible CRTC 0 et 1, que la même plus impressionnante seulement compatible CRTC 1. Avec cette même logique, on les invite donc à faire une démo encore moins impressionnante compatible CRTC 0, 1 et 2, plutôt que la même sur CRTC 0 et 1 !



The One (Impact)

Plus récemment, la partie finale de *Onescreen Colonies #3* s'ouvre sur un « CRTC 1 Only » qui a pu émouvoir les plus sensibles. C'est que, comme *Back To Futurs'*, elle exploite la possibilité d'une pseudo ligne à ligne avec R9 = 7, qui permet de déformer une image complète au pixel mode 0 sur tout l'écran, en jouant un sample à 16 kHz. Pour obtenir l'équivalent sur CRTC 0, il faut une RVI, qui consomme beaucoup plus de temps machine, ce qui impliquerait de retirer le sample.

DES DÉMOS PLUS SIMPLES SUR CRTC 1 ?

Des CPCistes mal informés me soufflent que *From Scratch* serait plus facile à faire sur CRTC 1 que sur 0. Examinons la chose. La démo comporte un plasma, un wobbler, un gros scroll, des petits scrolls et la 3D finale. Les parties en rupture classique (gros scroll) et ligne à ligne (plasma et 3D de fin) ne pré-



OPTIMISER SON BC26

Dans cet article, on s'interroge sur les diverses manières de descendre d'une ligne sur l'écran, ou comment optimiser son BC26.

PAR HICKS/VANITY

Beware! Ne pouvant tout reprendre, je supposerai que la structure de la mémoire vidéo (VRAM) du CPC est connue (voir par exemple *Quasar CPC 3 et 4*, ou *Quasar Net*). Vous trouverez dans l'encadré ci-contre quelques rappels relatifs au vocabulaire technique évoqué ici, si vous n'avez pas fait CRTC seconde langue au lycée.

« Or, donc », comme aimait à le répéter OffseT dans *Quasar*, sur CPC la VRAM n'est pas linéaire. Si c'était le cas, il suffirait d'ajouter la largeur de l'écran à l'adresse courante pour descendre d'une ligne. On écrirait simplement :

```
ld bc,R1*2
add hl,bc
```

Si vous faites ça sur CPC, vous descendrez de 8 lignes, ou plus exactement de $R9 + 1$ lignes. On se représentera donc plus précisément la VRAM du CPC comme un feuilletage de $R9 + 1$ couches. Afin d'éviter aux codeurs de s'arracher le peu de cheveux qu'il leur reste (selon l'âge), le système propose le vecteur SCR NEXT LINE, localisé à l'adresse &BC26 : on l'appelle avec HL contenant l'adresse courante, qu'il remplace au retour par

UN PEU DE VOCABULAIRE

- **R1** : registre 1 du CRTC, indiquant la position du border et par extension la largeur de l'écran en words. Un compteur interne au CRTC (C0 ou HCC, selon la chapelle) est incrémenté de 0 à $R1 - 1$ avant d'activer le border. Par défaut, $R1 = 40$,
- **R9** : registre 9 du CRTC, fixant le nombre de lignes élémentaires de chaque ligne caractère. Un autre compteur interne au CRTC (C9 ou VLC) évolue ici de 0 à $R9$ avant de passer à la ligne caractère suivante, c'est pourquoi avec $R9 = 7$ par défaut, les caractères font 8 lignes.

l'adresse de la ligne suivante. Par extension, un « BC26 » est devenu le nom du petit morceau de code qui permet de descendre d'une ligne, même sans utiliser le système.

Mais le BC26 du système comporte de trop nombreux défauts pour une programmation avancée : il est lent (un peu plus de 100 NOPs), suppose le système actif ainsi qu'une configuration CRTC standard.

Je vous propose ici un petit éventail de BC26 permettant de se passer du vecteur SCR NEXT LINE. Toutes n'imposent pas les mêmes contraintes et ne répondent pas aux mêmes besoins selon que vous voudrez afficher un sprite dans un jeu, un effet très rectangle dans une démo, des formes géométriques (cercle, triangle), etc. À vous de vous inspirer de la routine la mieux adaptée à vos projets.

TECHNIQUE 1 : LE CLASSIQUE

Débutons par la version la plus classique :

```
ld bc,&800
add hl,bc
jr nc,no_bloc0
ld bc,&c000+R1*2
add hl,bc
no_bloc0
```

On passe d'une ligne à l'autre en ajoutant &800 à HL, jusqu'au positionnement du flag Carry, qui nous indique que nous avons atteint le dernier bloc, et qu'il faut donc ajouter une autre valeur pour reboucler sur le bloc 0 de la ligne caractère suivante. C'est le rôle du $\&c000 + R1 \times 2$.

Si BC et DE ne sont pas utilisés, on peut optimiser cette première routine en les préchargeant respectivement avec &800 et $\&c000 + R1 \times 2$, et ainsi se limiter à :

```
add hl,bc
jr nc,no_bloc0
add hl,de
no_bloc0
```

L'avantage de cette technique est qu'elle est facile à écrire, pour un affichage standard qui ne demande pas à être optimisé. Les inconvénients sont cependant nombreux : elle est lente (6 à 8 NOPs), occupe pas mal de RAM si elle est déroulée (4 octets par



École Buissonnière (Overlanders)

ligne), sa durée fluctue à cause du saut conditionnel, elle n'est pas générique (ne fonctionne que sur la page &c000), et elle contraint à sacrifier plusieurs registres 16 bits.

J'ai écrit que la routine n'était pas générique. En effet, pour passer du dernier bloc au bloc 0, vous pourrez difficilement détecter le dépassement quelle que soit la page sans faire exploser le temps machine : il faudra une routine différente par page. Veuillez zieuter l'encadré en bas de cette page pour de plus amples précisions sur ce point.

TECHNIQUE 2 : AVEC LA PILE

Allons plus loin avec notre amie la pile, que nous ferons pointer sur une table d'adresses écran de début de ligne. Si BC contient la position X où débiter l'affichage, notre routine se limitera à :

```
pop hl      ; adresse de début de ligne
add hl,bc   ; + position x
```

Ainsi, le BC26 redevient générique (à condition d'avoir une table d'adresses par page), le temps machine est stabilisé, et le code ne prend plus que 2 octets. Inconvénients : il reste assez lent (6 NOPs) et le recours à la pile oblige à couper les interruptions, qui pourraient corrompre la table d'adresses. On pourrait cependant étudier dans un prochain article comment faire cohabiter harmonieusement pile et interruptions.

GESTION DU DÉBOREMENT D'UN BC26 CLASSIQUE (TECHNIQUE 1)

Page	Bits de la page	Code
&0000	&00 - &40 %00xxxxxx - %01xxxxxx	bit 6,h : jr z,no_bloc0
&4000	&40 - &80 %01xxxxxx - %10xxxxxx	bit 6,h : jr nz,no_bloc0
		bit 7,h : jr z,no_bloc0
&8000	&80 - &c0 %10xxxxxx - %11xxxxxx	bit 6,h : jr z,no_bloc0
&c000	&c0 - &00 %11xxxxxx - %00xxxxxx	jr nc,no_bloc0

Si vous commencez toujours à la même position X, vous pouvez aussi construire votre table d'adresses en incluant directement l'addition. Il ne resterait alors plus qu'à faire un simple POP HL de 3 NOPs pour descendre à la ligne suivante. C'est par exemple ce qui se passe lorsqu'on veut afficher des effets dans une fenêtre écran prédéfinie comme dans la démo *École Buissonnière*.

TECHNIQUE 3 : SANS CONDITION

Finalement, ce qui alourdissait la technique 1 était le test de débordement, qui n'est utile qu'une fois sur 8. Tentons d'éjecter le saut conditionnel (jr nc,no_bloc0) et de précharger BC avec &800 et DE avec &c000 + R1 × 2. Si on démarre sur le bloc 0, on sait qu'il suffit de faire :

```
7 ** add hl,bc ; passe aux blocs 1 à 7
    add hl,de ; passe au bloc 0
```

Si on démarre sur le bloc 1, il faudra 6 ADD HL,BC avant le ADD HL,DE, et ainsi de suite. Le BC26 devient alors simplissime, ne consomme qu'un octet, 3 NOPs, et est stable en temps machine. Inconvénient de taille : il devient nécessaire de dupliquer la routine d'affichage R9 + 1 fois, puisque la combinaison des ADD dépend de la ligne de départ.

TECHNIQUE 4 : 8 BITS SUFFISENT

La même technique peut être améliorée : ajouter &800 à HL revient finalement à additionner 8 à H, ce qui permet d'optimiser 7 des 8 routines :

```
7 ** [ld a,h
    add b      ; b = 8
    ld h,a]
    add hl,de
```

On ne mobilise plus C mais A, le temps machine reste le même, il est encore stable, mais la routine consomme deux octets de plus. « Quel est donc l'intérêt de cette routine ? » demande alors l'insolent petit chauve à lunettes (ex-petit blond à lunettes, qui a pris un coup de vieux, comme tout le monde). C'est qu'elle donne naissance à la variante suivante : puisque l'addition transite par A, ce registre contient déjà la valeur de H, ce qui permet de se passer du LD A,H la plupart du temps. On peut donc faire :

```
ld a,h      ; utile car le add hl,de qui
            ; précède peut modifier H
add b
ld h,a      ; passage du bloc 0 à 1
6 ** [add b  ; A contient déjà H
    ld h,a]  ; passage aux blocs 2,...,7
add hl,de   ; passage au bloc 0
```

On peut ainsi atteindre 2 NOPs dans 6 cas sur 8 (3 NOPs sinon), soit $3 + 6 \times 2 + 3 = 18$ NOPs pour 8 lignes, ou une moyenne de 2,25 NOPs par ligne. Avantage de taille : on descend de façon linéaire, en enchaînant les blocs dans l'ordre.

TECHNIQUE 5 : GRAY CODE ET ZIGZAG

Raffinons encore un peu le principe précédent. Oubliions les additions au profit d'opérations positionnant directement les bits des registres (n est le numéro du bit à positionner à 1 (SET) ou à 0 (RES)) :

```
7 ** set / res n,h
    add hl,de
```

Cette fois, c'est toujours 2 NOPs 7 fois sur 8, et 3 dans le dernier cas, ce qui donne 17 NOPs pour 8 lignes, ou 2,125 NOPs de moyenne. Il y a cependant deux problèmes techniques provoqués par l'usage des SET/RES, mais qu'on peut facilement contourner :

- puisque ces instructions ne modifient qu'un seul bit à la fois, elles n'ajoutent pas toujours &800 à l'adresse courante (valeur du bit 3 de H) et ne permettent plus de parcourir la mémoire linéairement. On aura alors recours à un Gray code, un codage binaire permettant de parcourir un intervalle (ici [&c0 ; &f8] avec un pas de 8) en ne modifiant qu'un seul bit à la fois.
- ces instructions permettent de descendre d'une ligne, mais sans retour à la ligne. Si vous copiez du bitmap de la gauche vers la droite (INC), puis exécutez un SET 3,H en bout de ligne, vous vous retrouverez à la fin de la ligne suivante, et il faudra réafficher de la droite vers la gauche (DEC). Il faut donc parcourir l'écran et votre bitmap en zigzag.

Vous trouverez dans l'encadré suivant un exemple schématique parcourant 8 lignes selon ces deux techniques.

ORGANISATION D'UN GRAY CODE AVEC ZIGZAG (TECHNIQUE 5)

Adresse	Bits 3 à 5	Zigzag	BC26
&c000	%xx000xxx	inc hl	set 3,h
&c800	%xx001xxx	dec hl	set 4,h
&d800	%xx011xxx	inc hl	res 3,h
&d000	%xx010xxx	dec hl	set 5,h
&f000	%xx110xxx	inc hl	set 3,h
&f800	%xx111xxx	dec hl	res 4,h
&e800	%xx101xxx	inc hl	res 3,h
&e000	%xx100xxx	dec hl	add hl,de
			de = &e000+R1*2
&c000+R1*2	%xx000xxx	...	

Cette version a l'avantage d'être rapide (2 NOPs), stable en temps machine et permet plus facilement d'utiliser d'autres registres que HL. En revanche, elle convient mal pour le tracé d'une forme géométrique car elle brise la continuité des tracés avec les sauts de ligne, elle implique d'avoir $R9 + 1$ fois la routine ce qui prend de la place (une astuce consiste à modifier R9), et à modifier la structure de ses bitmaps pour qu'ils encaissent le Gray code et le zigzag.

Au passage : on entend souvent que la mémoire non-linéaire du CPC complique les choses, ralentit l'affichage, etc. Balivernes, vous le voyez ! Si tel était le cas, passer d'une ligne à l'autre impliquerait par exemple d'additionner toujours 80 à l'adresse courante, ce qui éviterait certes la prolifération de routines en fonction de l'adresse de départ (souvent $\times 8$), mais ne nous offrirait pas le bonheur d'un $+800$ de 2 NOPs.

TECHNIQUE 6 : REGISTRES D'INDEX

Un BC26 de 0 NOPs, ça vous dit ? Quasi. Grâce aux registres d'index, on peut avoir un adressage indexé sur un intervalle de $[-128 ; +127]$. En fonction de la largeur de l'écran, on peut donc écrire sur plusieurs lignes d'un même bloc sans modifier IX, et même ajuster légèrement la descente sans INC ou DEC supplémentaires. Le soucis évident est qu'on affiche donc toutes les $R9 + 1$ lignes, et qu'il faut traiter les blocs un par un.

Par exemple si $R1 = 48$, on a -96, +96, ça passe. Il faut simplement additionner $R1 \times 2 \times 3$ à IX de temps en temps. Comme tout cela n'est pas multiple de 8, alors ça boucle moyennement, problème que je vous laisse régler car j'ai déjà les miens !

```
ld (ix-R1*2),d8
ld (ix),d8
ld (ix+R1*2),d8
add ix,bc
```

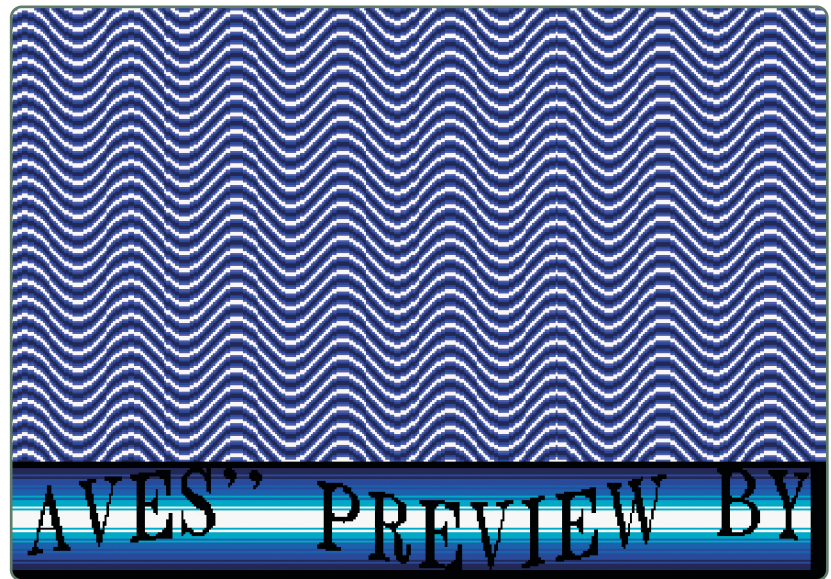
Au niveau temps machine, ce n'est pas folichon : 5 NOPs par LD, et encore 5 pour le ADD, soit $4 \times 5 = 20$ NOPs pour 3 lignes, donc $20 / 3 = 6,5$ NOPs par octet écrit (les précédentes estimations n'incluaient pas la lecture / écriture). On gagne cependant une certaine souplesse (on est pas obligé de descendre verticalement) et ne consomme pas trop de registres (HL et DE sont libres).

TECHNIQUE 7 : LE CRTC S'EN MÊLE

D'habiles demomakers ont imaginé le BC26 ultime, en 1 NOP seulement :

```
inc h
```

C'est tout ? Oui. La technique est probablement inaugurée par la preview *Waves* d'Overflow : avec une RVI, celui-ci structure sa mémoire écran de sorte à ce que chaque ligne débute à une adresse distante de 800 de la précédente, ce qui permet de passer d'une ligne à l'autre via un simple INC H. Très utile dans le cadre d'un sinus scroll, affiché en colonne où il faut faire $X \times Y$ BC26 !



Waves (Overflow)

Une RVI permet d'afficher sur la hauteur de notre choix, mais consomme presque la totalité du temps machine sur cette hauteur. Une alternative consiste à faire la même chose en ligne à ligne : dans ce cas, vous n'aurez accès qu'à 8 lignes par page VRAM, soit 32 au total. Limite contournable !

On a ainsi un BC26 ultra-rapide, compact (1 octet), au temps machine stable et qui ne nécessite pas de dupliquer 8 fois la routine. Quelques contraintes : il demande du temps pour la gestion CRTC (RVI ou LAL), implique une adaptation selon le CRTC (si RVI) et ne convient pas pour une zone trop haute (si LAL).

CONCLUSION

Si vous connaissez une 8^e manière de faire un BC26 que vous jugez digne d'intérêt, n'hésitez pas à l'envoyer à la rédaction et nous la publierons dans le prochain numéro ! ■



SAMPLEUR ET SANS REPROCHE

Me revoilà pour un article qui pourra intéresser les amateurs de Lo-Fi dans une utilisation détournée de notre cher AY, loin des sons carrés habituels.

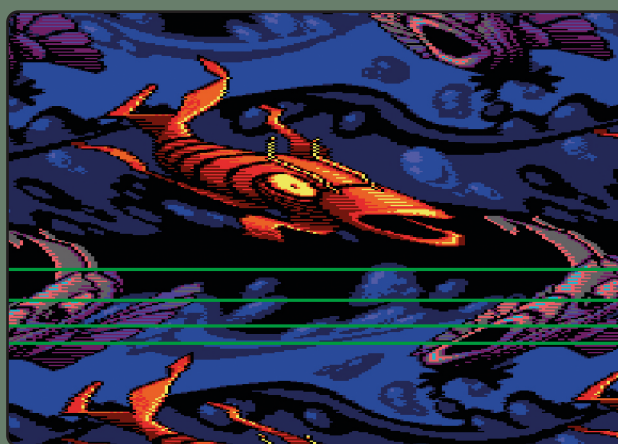
PAR ZIK/FUTURS'

La seconde partie de la démo *Onescreen Colonies #3* de Vanity réalise la prouesse d'accompagner un effet visuel en plein écran, fort esthétique, d'une bande audio en sons échantillonnés (audio samples). Cet article va tenter d'expliquer comment je suis arrivé à ce résultat sonore en commençant par la conversion des samples, puis en montrant comment en créer une séquence. Ma méthode en tout cas.

CONVERSION

Le générateur sonore AY-3-8912 du CPC n'est pas prévu pour jouer des samples. Cependant, même lorsque les canaux audio sont fermés, la programmation du registre de volume est prise en compte. L'AY présente en sortie un niveau continu qui correspond à la valeur de volume du canal. En reprogrammant le registre de volume rapidement, à une fréquence qui correspond au taux d'échantillonnage, on peut donc jouer un sample. Ouf !

Mais évidemment les limitations du AY s'appliquent : le volume est seulement sur 4 bits (16 valeurs) et les niveaux de sortie sont logarithmiques (plus ou moins). On devra donc faire pas mal de compromis au niveau du rendu sonore. Pas ouf.



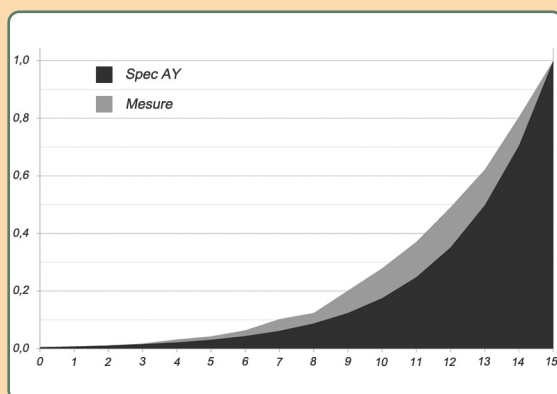
Onescreen Colonies #3 (Vanity)

Tout ceci étant posé, comment faire au mieux pour rejouer un sample avec une qualité correcte voire optimale ?

Ça paraît idiot, mais ça commence par le choix des samples ! Les sons saturés ou bruités, une guitare électrique ou une caisse claire, tolèrent mieux la conversion. Mais un son très pur et cristallin subira d'atroces sévices à cause de la trop faible résolution de sortie, notamment dans les valeurs de volume

élevées (cf. l'encart sur la courbe AY). Ceci dit, vous pouvez toujours tenter la conversion et juger du résultat, on a parfois de bonnes surprises. Un son peut être intéressant même s'il s'éloigne de celui d'origine.

COURBE DE VOLUME DU AY



Les spécifications du AY montrent une courbe de niveau de sortie parfaitement logarithmique avec un rapport 1,2 / 1,7 entre un volume et celui inférieur. En pratique ça n'est pas aussi parfait et il y a probablement aussi de légères variations d'un lot de composants à l'autre. MAME semble être très proche de cette courbe qui a pour caractéristique de sortir la moitié du niveau maximum pour un volume à 13. Grim et moi avons fait des mesures sur CPC qui montrent des courbes similaires entre elles mais un peu moins creusées que la courbe théorique, le volume de demi-niveau est à 12. Je n'ai jusqu'ici pas constaté une différence de qualité de conversion radicale selon la courbe utilisée.

Une fois votre son choisi, la première étape est de rééchantillonner le sample pour que sa fréquence corresponde à celle à laquelle on va le jouer, sinon sa hauteur et sa vitesse seront modifiées. Sauf exception, il ne s'agit pas d'une simple interpolation et il faut utiliser un logiciel adéquat pour cette opération. Vous pouvez en profiter pour le passer en mono dans la foulée.

Ensuite on en arrive aux données elles-mêmes. Elles ont une résolution bien supérieure à ce que l'on peut reproduire, le plus souvent un sample WAV utilise des valeurs entières signées au moins sur 16 bits. C'est là que l'on perdra le plus en qualité.

L'idée qui vient à l'esprit en premier est de passer directement en 4 bits. On peut le faire simplement, en inversant le bit de poids fort pour transposer la va-

leur en non-signé et en ne gardant que les 4 bits de poids fort, ce qui correspond à une division, avec en plus un arrondi si vous voulez. Et franchement dans certains cas ça donne de bons résultats.

Mais ce faisant, on a diminué la résolution comme si l'échelle d'arrivée était linéaire, ce qui n'est pas le cas. Pour réduire la distorsion, on peut tenir compte de la propriété logarithmique des volumes AY dans cette conversion.

Ces deux méthodes de conversion ont des inconvénients communs. D'une part, la valeur de repos est non-nulle (elle est au milieu de la plage de sortie). Or ça n'est pas idéal, le AY génère des sons carrés en oscillant entre 0 et le niveau de volume programmé. Si on joue une musique qui alterne son et sample on aura souvent des « pops » aux transitions.

D'autre part, ça oblige en définitive à jouer des samples à fort volume pour améliorer leur qualité en utilisant la plage maximale. Imaginez un sample à volume réduit, il n'utilisera que la plage de volume AY autour de la valeur médiane (en niveau), où la résolution est déjà très pauvre (cf. la forme d'onde B en illustration).

A contrario, dans la partie de fin de la démo CRTC³ de roudoudou, la musique et les samples ont un volume modéré par rapport à ce qui précède, il n'est pas possible d'arriver à ce résultat avec les méthodes ci-dessus.



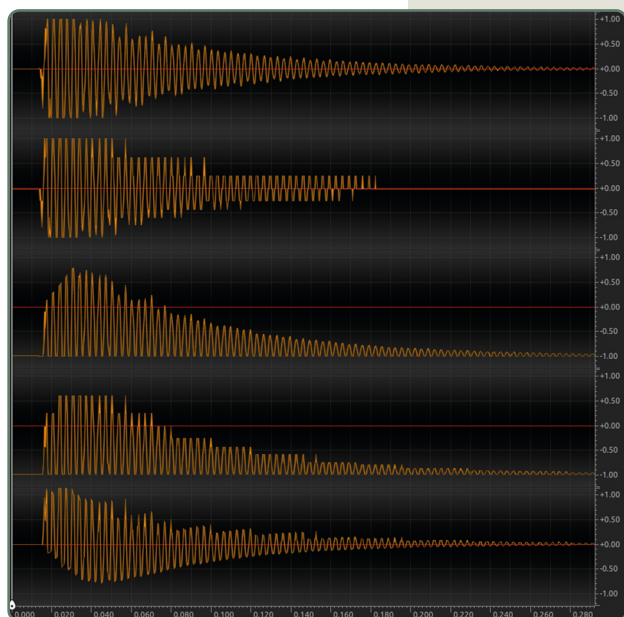
CRTC³ (Flower Corp. & Futurs' & X-men)

Je propose donc d'aller plus loin en prenant avantage de la nature logarithmique du volume AY qui fait que les pas sont petits pour les faibles volumes et de plus en plus grands en augmentant. Autrement dit, pour les volumes élevés la résolution est lamentable, les variations dans cette zone produisent un bruit de quantification important. En revanche pour les volumes faibles on s'approche de la résolution d'un sample 7 ou 8 bits.

L'idée ici est que, si le sample n'utilise pas toute la plage dynamique en permanence, on peut le décaler dans ces moments-là vers les valeurs plus petites de volume où la résolution est meilleure. Attention tout de même, il s'agit de ne pas décaler trop vite, sinon on superposera un son audible.

Pensez au son d'une cymbale, d'une cloche, d'un piano. Tous ces sons sont forts à l'attaque puis le volume sonore décroît progressivement. Si on garde les méthodes décrites précédemment, la fin du son sera un mélange de « crrr » et de « fffff ». Si on arrive à appliquer cette dernière méthode correctement, on maximise la résolution au fur et à mesure que le son évolue. La conséquence bonus naturelle est que le son s'éteint à 0.

Vous n'avez rien compris ? Voici une illustration.



A : forme d'onde du son d'origine.

B : conversion directe en niveaux AY centrée sur le niveau moyen en volume, la résolution dans cette zone distord le son et ne permet pas de suivre son affaiblissement.

C : traitement intermédiaire : le son d'origine est décalé vers les valeurs faibles selon sa dynamique (on peut noter que ma routine de traitement n'est pas parfaite, notamment à l'attaque du début). Cette forme d'onde sonne théoriquement comme la A, on n'a pas encore diminué la résolution.

D : le son C converti en niveaux AY. Notamment, la queue du son est ici préservée. La chaîne audio de sortie va supprimer les basses fréquences inaudibles et recentrer la forme d'onde (E).

En pratique, j'applique souvent quelques traitements avant la conversion pour en améliorer le résultat, des traitements de la dynamique comme du gain et de la compression. Pour faciliter les manipulations, j'en réalise une bonne partie dans un logiciel de production audio, *Reaper* en l'occurrence. En plus d'être bien adapté à ce genre de travail, il permet d'écrire ses propres plug-ins d'effet JSFX (rien à voir avec Javascript) en langage de script EEL2, et on peut s'inspirer des effets scriptés installés par défaut. J'ai développé deux plug-ins d'effet audio, un premier qui en fonction de la dynamique du son décale la forme d'onde comme expliqué plus haut et un second qui applique une quantification avec les niveaux logarithmiques du AY. Grâce à ce dernier, je peux entendre directement dans *Reaper* ce que donnera à peu près le son une fois joué par le CPC et éventuellement faire plusieurs essais de retouches.

Quand je suis content du résultat, je peux exporter la sortie du premier plugin en WAV et le passer dans un programme qui va appliquer la quantification des volumes AY et générer le fichier à jouer sur CPC. Et voilà !

SÉQUENCEUR

L'objectif pour la démo n'était pas de jouer un enchaînement de longs samples complexes en grosses sections (intro, couplet, refrain, par exemple) comme c'est fait dans les écrans d'introduction du *Manoir de Mortevielle* ou *Crazy Cars II*. Il s'agissait de mettre en place un séquenceur, une espèce de boîte à rythme sur un canal mono. C'est-à-dire qu'ici, l'idée est de découper le temps en tranches et d'avoir un jeu de samples unitaires d'une durée multiple de cette tranche de temps (unitaires mais éventuellement polyphoniques, comme un accord de plusieurs notes ou un mélange de sons). Cette façon de faire permet de garder une empreinte mémoire totale modeste, ce qui était une des contraintes fortes.

Le temps machine étant compté lui aussi, il n'est pas question de faire des calculs pour modifier la hauteur des sons ou autre, tout au plus décompresser les données des samples à la volée.

Pour trouver l'inspiration je suis parti de courtes boucles de batterie toutes faites (qu'on peut trouver ici ou là) en portant attention au style, au tempo, au nombre de sons qui permettraient de la recomposer, à la qualité qu'on peut espérer obtenir après conversion. Dans un deuxième temps, j'ai reconstitué une boucle allongée par des variations, puis ajouté d'autres sons complémentaires (scratches, accords). Tout est possible tant qu'il reste de la RAM et que l'inspiration est là !

L'effet visuel prenant tout l'écran, la gestion de la séquence n'a lieu qu'une seule fois par image (à 50 Hz, soit toutes les 20 millisecondes) et doit durer le minimum de temps possible. Elle est aussi écrite de façon à prendre un temps d'exécution constant. Cette gestion détermine quoi jouer pendant les 20 ms à venir : soit on démarre un nouveau sample, soit on continue de jouer le sample courant. Pour en limiter le temps machine, j'ai décidé que la description de la séquence serait une suite de couples de valeurs [pointeur, durée]. Le pointeur indique l'adresse des données de sample, la durée est le nombre d'images avant de passer au pas de séquence suivant. La valeur 1 comme durée de pas est utilisée en tant que telle mais elle est aussi un marqueur de fin pour la gestion du bouclage, j'aurais peut-être pu faire mieux au niveau temps machine. En revanche le fait d'avoir une petite intro avant la séquence qui boucle ne coûte rien, c'est simplement une valeur d'initialisation du pointeur de séquence, aucun temps machine n'est associé.

Pour éviter d'avoir à traiter un cas particulier, produire un silence se fait en jouant un sample vide, quitte à sacrifier un peu de mémoire.

Pendant l'affichage de l'effet, il s'agit uniquement d'envoyer une valeur par ligne dans le registre de volume du canal central du AY et d'incrémenter le pointeur. La sélection du registre AY a été faite une fois pour toute dans la mise en place de la démo, on a donc seulement à envoyer la valeur, mais ça reste assez coûteux en temps machine sur CPC Old parce qu'on accède au AY au travers du PPI.

Cet envoi doit se faire avec autant de régularité que possible, toutes les 64 μ s. Vu le résultat recherché, le choix d'une fréquence d'échantillonnage à 15625 Hz s'est imposé (ça correspond à un échantillon par ligne écran balayée), le multiple inférieur à 7,8 kHz coupe trop les aigus (théorème de Shannon, tout ça).

Le fait que la gestion n'ait lieu qu'une fois par image impose des contraintes sur la longueur des samples et sur le tempo de la séquence. Il faut que les samples aient une longueur multiple de 312 échantillons (20 ms) puisqu'aucune gestion de fin de sample n'est prise en charge. En réalité 312 lignes est un pire cas, en pratique il y a quelques lignes par écran pendant lesquelles on ne joue pas le sample (il faut bien évidemment gérer la séquence et l'effet visuel). Mais il est important que le nombre de lignes de gestion sans envoi de donnée de sample reste

minimal, chaque ligne ajoutée dégrade un peu plus la qualité audio. Pour ces lignes sans envoi de donnée deux choix sont possibles : soit on saute les données non jouées en incrémentant le pointeur, soit on ne fait rien. Dans le premier cas, on maintient la hauteur des sons mais on risque de créer des discontinuités dans la forme d'onde, dans le deuxième cas on baisse la hauteur des sons, ce qui n'était pas trop gênant dans le cas présent. C'est la deuxième option qui a été retenue pour la meilleure qualité du résultat.

J'ai choisi un tempo de 125 BPM à la noire avec une subdivision à la double-croche, l'unité de temps la plus courte de ma séquence et donc mes samples. La gestion à la VBL a aussi une incidence sur le tempo – c'est la même problématique pour toute musique AY avec un player à 50 Hz – toutes les valeurs ne sont pas possibles, on doit choisir un tempo parmi une liste de valeurs dont 94, 107, 125, 150, 187, etc.

Une noire à 125 BPM, ça veut dire qu'elle dure 480 ms. La décomposition à la double-croche est donc à 120 ms = 6×20 ms = 6 frames. Tous les

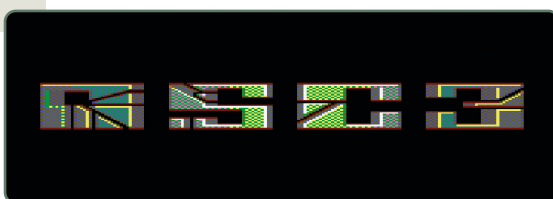
samples ont donc une longueur multiple de 120 ms, d'où les $6 \times n$ que vous trouverez dans les données de séquence dans l'extrait de code source (avec n de 1 à 3). Puisque vous voulez tout savoir, je

l'avoue, un des samples dure $6 \times 2,5$ mais c'est une optimisation de l'empreinte mémoire, il est immédiatement suivi d'un silence de durée $6 \times 0,5$. Le compte est bon !

Pour continuer dans les chiffres, ce que vous entendez dans la démo est composé à partir de 12 samples (4 percussions, 2 accords, 6 scratches), plus un de silence. Ils occupent environ 44 ko en mémoire puisqu'ils sont en définitive stockés non compressés pour cause d'optimisation d'1 NOP non trouvée !

Pour travailler la séquence j'ai utilisé un autre logiciel de production audio – littéralement, un séquenceur – avec un plugin instrument genre sampleur ou drum machine dans lequel j'ai assigné chaque sample à une note de la gamme. Une fois la boucle composée, il suffit de la transcrire dans les données de séquence du code source et le tour est joué !

Les samples sont assez rares sur CPC, ils sont gourmands en mémoire et en temps processeur, mais avec quelques précautions on arrive à un résultat satisfaisant qui mérite ces efforts. Au plaisir d'écouter vos productions ! ■



```

; Gestion de la séquence de samples
; appelée une fois par écran
; sortie ; BC pointe vers le sample à jouer
SmpSeqFn
    ld hl,(SeqPtr)
SSFCnt ld a,1
    dec a
    jr nz,SSF_tr1
    ld c,(hl)
    inc hl
    ld b,(hl)
    inc hl
    ld a,(hl)
    inc hl
    cp 1 ; gère le bouclage de la séquence
    jr nz,SSF_tr2
    nop
    ld hl,Seq_start
SSFn1 ld (SSFCnt+1),a
    ld (SeqPtr),hl
    ret

SSF_tr1 ds 19-3
SSF_tr2 jr SSFn1

SeqPtr dw Seq_intro ; pointeur vers le pas
        ; courant de la séquence

SmpSeqInit
; prépare l'envoi du sample
; pas de bruit et canaux fermés
    ld c,7 ; registre
    ld a,&3f ; valeur
    call Send2PSG
; sélectionne le registre 9 (vol B)
    ld c,9 ; registre
    xor a ; valeur
    jp Send2PSG

; pointeur vers le sample, nombre de frames
Seq_start
    dw smp1:db 6*2
    dw smp3:db 6*1
    dw silence:db 6*1
    dw smp2:db 6*2
[...]
    dw smp2:db 6*2
    dw silence:db 6*1
Seq_intro
    dw smp3:db 6*1
    dw silence:db 6*1
    dw smp2:db 6*3
    dw silence:db 6*1-1
;
    dw silence:db 1 ; fin de séquence
Seq_end

```

ERRATUM! (HICKS/VANITY)

Nom d'un concombre kaki ! Le premier numéro de *64 NOPs* contenait des informations inexactes, voire erronées ! Nous étions encore jeunes, pardonnez-nous...

Un certain Hicks (qui a été renvoyé depuis), prétend que les splits de 3 NOPs de l'introduction de *KKB First* ne sont en fait que du border. Faux ! Un réexamen attentif montre que c'est encore plus intéressant, car plus facile à exploiter. Tom et/ou Chilly (on hésite) insèrent en réalité dans une série de OUTI un OUT (c), r8 classique (où r8 désigne tout registre 8 bits, comme C, par exemple). Mais alors ?! La valeur envoyée par un OUTI est prise en compte lors de la 5^e et dernière NOP, alors que celle d'un OUT (c), r8 l'est à la 3^e NOP. Si on alterne OUTI et OUT (c), r8, on se retrouve donc avec des bandes de 3 NOPs (couleur associée au OUTI) et 6 NOPs (couleur associée au OUT). On voit donc qu'en plaçant un simple OUT au milieu des OUTI, les KKB obtiennent ce split raster de 3 NOPs, sans recours à l'instruction OUT (n), a ni au border ! Erreur de votre serviteur, donc, mais erreur féconde : cette fausse piste a inspiré le logo « Vanity » qui introduit *Onescreen Colonies #3*, et qui comporte donc de ce fait le premier graph de l'histoire intégrant astucieusement du border pour augmenter le nombre de couleurs !



Il n'est pas rare qu'un CPCiste ait l'impression de sincèrement découvrir une technique, qu'il ne fait en réalité que redécouvrir. Ainsi, suite à divers échanges sur le forum du CPCWiki, on apprend que le fameux RST #38 conditionnel imaginé par Golem13 et présenté dans le dernier numéro était déjà utilisé par le codeur de *Jet Set Willy II: The Final Frontier*, qui glisse dans ce même jeu un petit message :

« Well, here is a little message to those of you who dare to HACK into other peoples programs! I cannot be bothered to put line feeds or carriage returns into all this, so there. Interesting thing, the Z80. Did you know, for example, about the conditional RST #38 instruction? Just try doing a JR Z, -1. Any condition will work. Not much use on the Amstrad! What do you mean you use it all the time!?!?!? Bye, and good luck with the game (tee hee!) »



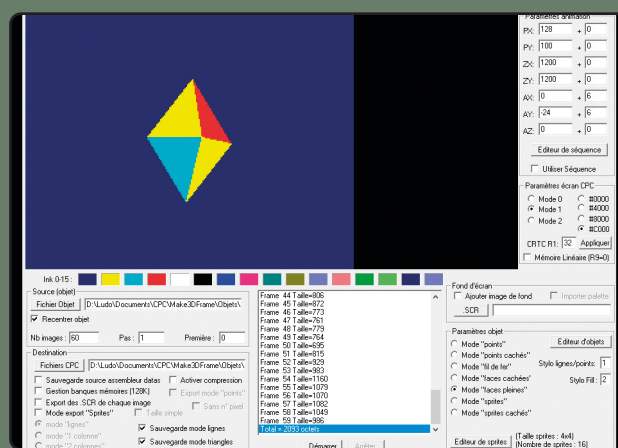
DES TRIANGLES SUR CPC

Un codeur a toujours dans ses archives des vieux bouts de code jamais diffusés, mais qui « peuvent servir ». Parfois, une idée permet à ces reliques de resurgir et de créer quelque chose d'intéressant. Ou une demande d'une autre personne permet de finaliser un projet endormi. Voici donc l'explication de la naissance de deux logiciels et deux démos, tous basés sur le même principe...

PAR DEMONIAK/IMPACT

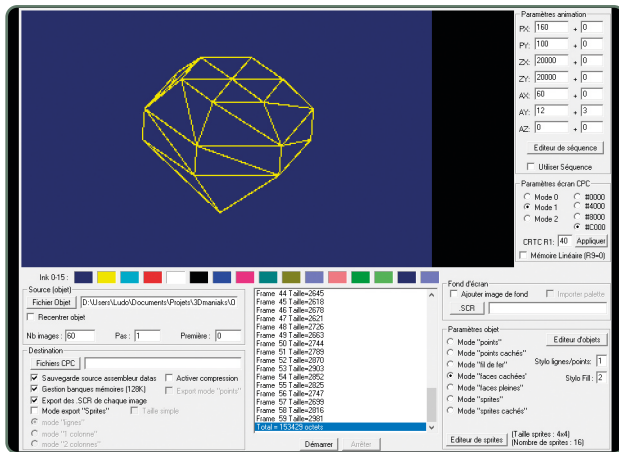
La majorité des logiciels de 3D utilisent des objets définis sous la forme de triangles. Par exemple, un cube, composé de 6 faces carrées, peut être représenté par 12 triangles (2 triangles par face). Il en va ainsi pour n'importe quel objet. Évidemment, si l'on veut obtenir des objets de meilleure définition, on définira beaucoup plus de triangles et de plus petite dimension. D'où l'idée du logiciel *Make3DFrames* qui, à partir d'objets « 3D » (définis avec des triangles), permettait de générer des animations sur le CPC. Ces animations étaient en fait de simples images affichées en delta-packing (différences inter-image). Il aurait été intéressant de pouvoir afficher directement ces objets 3D à l'aide d'une routine de tracé de triangles sur CPC... D'où l'évolution de *Make3DFrames* qui, dans sa dernière version, peut exporter une liste de triangles pour chaque image (avec coordonnées et couleurs).

J'ai ensuite fait quelques recherches sur le net pour trouver un algorithme de tracé de triangles rapide, pour pouvoir ensuite l'adapter au CPC. Celui que j'ai retenu est le suivant : on peut décomposer chaque triangle en deux, le premier ayant une base horizontale, qui sera l'un des côtés du second. En fait, c'est comme si l'on « coupait » le triangle avec une ligne



Make3DFrames

horizontale passant par l'un de ses sommets. Pour cela, il faut tout d'abord trier les coordonnées des 3 points du triangle suivant leur ordonnée (Y). En supposant que l'on nomme les coordonnées du triangle (X1,Y1), (X2,Y2), (X3,Y3), on « coupera » donc le triangle avec une droite horizontale passant par Y2. Ensuite, c'est assez simple. Nous avons donc deux triangles avec un côté horizontal. Il suffit de remplir ces deux triangles avec des lignes horizontales pour obtenir le résultat voulu.



Voici un exemple en C d'une fonction de tracé de triangles à l'aide de lignes horizontales :

```
void FillTriangle(int x1, int y1, int x2,
int y2, int x3, int y3, int color) {
    int dx1 = x3 - x1;
    int dy1 = y3 - y1;
    int sgn1 = Sign(dx1);
    dx1 = Abs(dx1);
    int err1 = 0;
    int dx2 = x2 - x1;
    int dy2 = y2 - y1;
    int sgn2 = Sign(dx2);
    dx2 = Abs(dx2);
    int err2 = 0;
    int x1 = x1;
    int xr = x1;
    if (y1 == y2)
        xr = x2;

    for (int y = y1; y < y3; y++) {
        DrawLine(x1, y, xr, y, color);
        err1 += dx1;
        while (err1 > dy1) {
            x1 += sgn1;
            err1 -= dy1;
        }
        if (y == y2) {
            // Tracé du second "demi-triangle"
            dx2 = x3 - x2;
            dy2 = y3 - y2;
            sgn2 = Sign(dx2);
            dx2 = Abs(dx2);
            err2 = 0;
        }
        err2 += dx2;
        while (err2 > dy2) {
            xr += sgn2;
            err2 -= dy2;
        }
    }
}
```

Cet algorithme se résumait à quelques instructions arithmétiques simples (additions, soustractions, comparaisons), mais utilisait la fonction `DrawLine(x1,y1,x2,y2,color)`. Cette fonction doit nous permettre de tracer une ligne horizontale d'ordonnée Y1 (Y2 = Y1) depuis l'abscisse X1 jusqu'à l'abscisse X2.

J'ai choisi de travailler en mode 1, car c'est le mode qui nous donne des pixels plus ou moins carrés. *A priori* tracer une ligne horizontale en mode 1 en assembleur peut sembler facile, mais on doit quand même se poser quelques questions :

- **Déterminer l'adresse de début de la mémoire vidéo.** Ceci peut se faire facilement, à l'aide de la formule suivante :

$$(X \gg 2) + (Y \gg 3) \times \text{NbCols} + (Y \& 7) \times \#800$$

avec X l'abscisse de départ, Y l'ordonnée, NbCols le nombre de colonnes caractères (équivalent à la valeur du registre 1 du CRTC multiplié par 2). À cette adresse il faut ajouter l'adresse de base de la mémoire vidéo, qui est #C000 par défaut sur le CPC.

- **Déterminer quel est le pixel de départ.** Oui, car en faisant $X \gg 2$ (ou $X / 4$), on obtient une adresse écran, mais un octet en mode 1 peut contenir 4 pixels. Notre pixel de départ correspond en fait au reste de la division entière de $X / 4$.
- **Déterminer quel est le pixel de fin.** Pareil que pour le pixel de départ, on doit savoir quand on s'arrête.
- **Choisir en fonction de la couleur les valeurs à écrire en mémoire écran.** Évidemment, tracer une ligne avec le stylo 1 sera différent de tracer une ligne avec le stylo 2 ou 3.

Prenons un exemple concret, je veux exécuter la fonction suivante : `DrawLine(2,100,10,100,1)`.

- **Je calcule l'adresse de départ.**

$$\begin{aligned} & (X \gg 2) + (Y \gg 3) \times \text{NbCols} + (Y \& 7) \times \#800 \\ &= (2 \gg 2) + (100 \gg 3) \times 80 + (100 \& 7) \times \#800 \\ &= 0 + 12 \times 80 + 4 \times \#800 \\ &= 9152, \text{ ou } \#23C0 \text{ en hexa.} \end{aligned}$$

Soit $\#23C0 + \#C000 = \#E3C0$ sur le CPC.

- **Je détermine le pixel de départ.**

$$X \text{ MOD } 4 = 2 \text{ MOD } 4 = 2$$

Le pixel de départ sera le numéro 2 (en numérotant les pixels de 0 à 3).

- **Je détermine le pixel d'arrivée.** Là c'est un peu plus complexe. Je dois tracer au total $10 - 2$ ($X_2 - X_1$) soit 8 pixels. Je commence au pixel numéro 2 sur le premier octet. Je vais donc dessiner 2 pixels (le 2 et le 3) sur le premier octet. Octet suivant, pas de soucis, je remplis les 4 pixels. Octet suivant, j'ai déjà tracé 6 pixels, il m'en reste donc 2.
- **Choix des octets à écrire en fonction de la couleur déterminée.** Je ne ferai pas ici un cours sur l'organisation de la mémoire vidéo du CPC en mode 1, mais un petit rappel :
 4 pixels en stylo 1 = #F0 (octet de VRAM)
 4 pixels en stylo 2 = #0F (octet de VRAM)
 4 pixels en stylo 3 = #FF (octet de VRAM)

Pas facile donc de faire une fonction universelle de tracé de ligne en mode 1. Sauf en utilisant une table de couleurs. En gros, une table indexée par le numéro de stylo qui contiendrait les octets à écrire en mémoire vidéo.

Voilà le principe général de cette fonction de tracé de ligne horizontale. Pour des questions de rapidité, plutôt que de calculer l'adresse mémoire écran, j'ai préféré passer par une table précalculée.

Une fois l'adaptation terminée en Z80 de ces routines, je n'ai malheureusement pas été plus loin pendant pas mal de temps... Mais un beau jour de l'année 2021, je me suis dit : et si j'essayais d'afficher un dessin à base de triangles ? Vu que ma routine était limitée à des coordonnées sur 8 bits (0 à 255), l'utilisation d'un écran carré (256 × 256 pixels) en mode 1 me semblait judicieux.

J'ai donc essayé de générer quelques images, tout d'abord « à la main » avec quelques triangles. Le résultat me semblait intéressant, mais il fallait trouver un moyen de générer plus facilement des images. Du coup, je me suis mis dans l'idée de développer un petit logiciel de dessin de triangles sur PC, qui permettrait de générer des images sur CPC avec ces triangles. J'ai appelé ce logiciel *Triangul'Art*.

Le principe de ce logiciel est assez simple, il permet de dessiner dans une fenêtre de 256 × 256 pixels une image composée de triangles. Une fois l'image terminée, on peut la sauvegarder (au format XML) et générer un source Z80, contenant simplement les coordonnées de chaque triangle composant l'image.

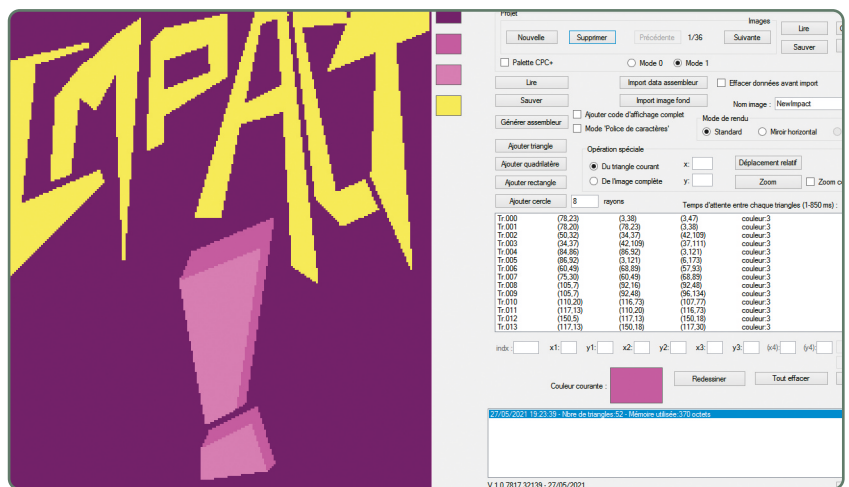
J'ai continué à développer le logiciel en ajoutant deux nouvelles fonctions intéressantes :

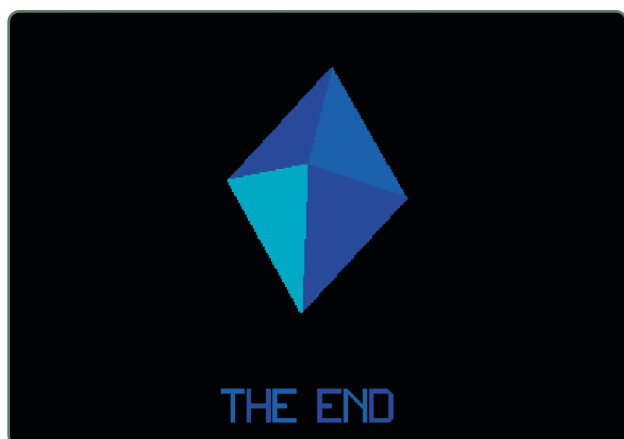
- Permettre de donner à un triangle un effet miroir (duplication du triangle soit en horizontal, soit en vertical). Cela permettrait de gagner en mémoire sur des images ayant une symétrie axiale.
- Une autre fonction intéressante est la possibilité d'importer une image de fond, pour pouvoir ensuite dessiner les triangles par-dessus.



Triangul'Art

Une fois ces fonctions terminées, l'idée de faire une démo à base d'images composées uniquement de triangles était lancée. J'ai donc dessiné pas mal d'images en format « triangles » pour la composition de la démo. La routine d'affichage de triangles en Z80 était assez rapide, mais j'ai pensé que ça serait sympa de voir l'image se dessiner triangle par triangle, d'où l'intérêt d'ajouter un paramètre de temps d'affichage de chaque triangle dans *Triangul'Art*.





Triangul'Art

La structure de la démo commençait à prendre forme. Il ne restait plus que la partie « greetings » à imaginer. L'idée m'est venue d'avoir une police elle aussi composée de triangles. C'est donc une option que j'ai ajoutée à *Triangul'Art* : permettre de générer un caractère à partir de triangles.

Les images étaient prêtes, la police aussi, le writter terminé, il me manquait quand même un petit « plus » pour finaliser cette démo... En effet, la suite d'images à afficher prenait un certain temps, mais trop court finalement pour une démo. Une idée m'est venue : pourquoi ne pas réafficher ces images, mais sans prendre en compte le paramètres de temps d'affichage des triangles. La routine de tracé de triangle étant assez rapide, ça aurait donné un effet plus dynamique à la démo. J'affichais donc chaque image le plus rapidement possible, pour montrer un peu la vitesse de cette routine.

Et j'ai eu l'idée d'ajouter à la fin une animation composée de plusieurs images, elles-mêmes composées de triangles. Et voilà la démo *Triangul'Art* (créée à l'aide du logiciel *Triangul'Art*) bouclée mi-mai 2021. :-)

Fin mai 2021. Au vu des commentaires sur Pouet, la partie finale 3D animée est ce qui a le plus plu dans la démo. Alors, avec mes collègues d'Impact, on se dit que ça serait bien de faire une démo uniquement composée d'animations 3D. Ni une ni deux, je me mets à chercher des objets 3D sympas mais pas trop volumineux pour essayer de les animer et voir le résultat. Je fais toujours ça avec mon bon vieux *Make3DFrames*, qui permet d'importer ces objets et de générer des animations (une liste de coordonnées de triangles pour chaque image).

C'est bien, mais je remarque que parfois il y a des triangles dessinés mais très peu visibles à l'écran. En les supprimant, on gagnera évidemment en temps d'exécution et aussi en espace mémoire, tout

cela sans perte de définition de l'objet. Je modifie le logiciel *Triangul'Art* pour qu'il permette de relire les fichiers d'animation générés par *Make3DFrames*, et permettre d'éditer chaque image. Une fonction « optimisation » est même ajoutée, qui permet de repérer les triangles très peu visibles dans une image. C'est un peu de travail manuel, mais le résultat est sympa.

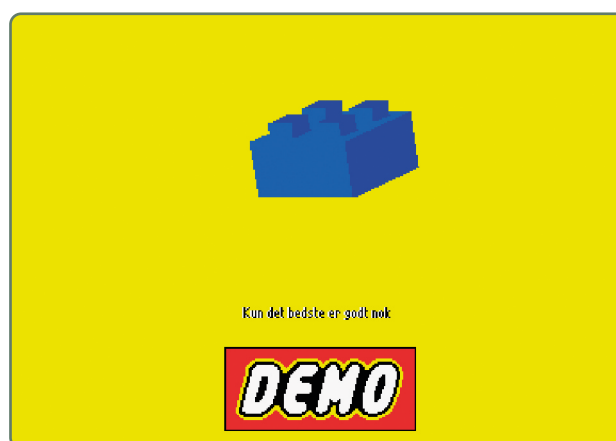
La routine qui affiche les triangles utilise bien sûr du double-buffering : on dessine par exemple en #8000 pendant que l'on affiche la mémoire en #C000, puis on inverse. Pour l'effacement, j'utilise la méthode suivante : je calcule le rectangle de mémoire vidéo qui sera utilisé pour l'affichage de l'animation complète, et c'est ce rectangle que j'efface à chaque fois avant de redessiner les triangles de l'image suivante.

Au départ, je pensais faire une démo qui tiendrait dans les 64K de n'importe quel CPC de base (*Triangul'Art* était une 16K). Mais cela limitait le nombre d'objets à afficher, et mes sympathiques collègues d'Impact m'ont convaincu de faire une démo tournant avec 128K de mémoire.

Les animations d'objets s'ajoutaient donc, et quelques idées viennent améliorer l'ensemble :

- Ajouter une image de fond à certaines animations (ces images de fond étant générées elles aussi avec des triangles).
- Utiliser une police vectorielle, permettant plusieurs tailles d'affichage de caractères.
- Ajouter quelques parties différentes des objets 3D pour un peu plus d'originalité.

Le code avançait plutôt bien, et je trouvais encore quelques optimisations spécifiques à la démo dans la routine de tracé de triangle. Par exemple, HL était un pointeur vers la table des adresses écran (table de 256 mots), et en plaçant cette table à l'adresse #700 (dans HL), j'ai pu remplacer quelques AND 7 et OR 7 par des AND H et OR H.



3DManiaks



En plus de la routine de tracé de triangle, j'ai ajouté également une routine de tracé de ligne. Cela permettait de faire quelques effets sympas : comme les 16 triangles en rotation, le masque d'*Iron Man*, et surtout le cœur à la fin de la démo, qui est affiché une fois en lignes, une fois en triangles.



Pour la musique, c'est encore un PT3 venant du ZX Spectrum qui fit l'affaire, nous n'avons malheureusement pas encore de musicien chez Impact...

On est fin juillet, il y a le meeting BND qui se rapproche, et on se dit que ça serait sympa de releaser la démo pour cette occasion. Début août, je profite qu'il me reste une semaine de congés pour aller passer quelques jours chez Kris. Bien sûr, on a bossé sur la démo, en améliorant quelques effets, notamment sur la partie graphique. CMP était avec nous sur les réseaux, et nous a également donné son avis et pas mal de conseils sur le design de la démo.

Restait à lui trouver un nom : *3DManiaks* fut validé par tout le monde (pourquoi ce nom ? 3D pour les objets, Maniaks un peu en référence à mon pseudo).

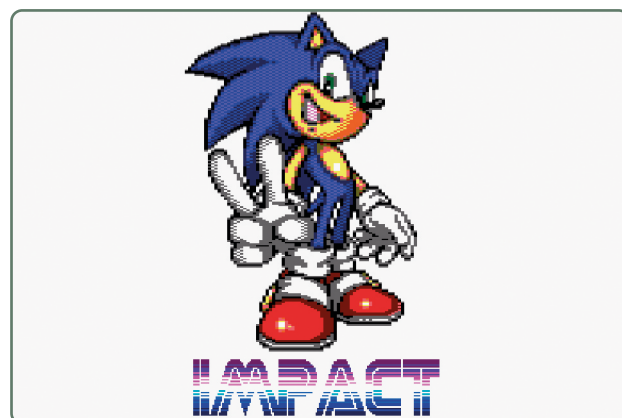
Ensuite, la démo faisant environ 100K de code et de données, il aurait été dommage d'avoir un simple lanceur BASIC qui ne faisait que charger les fichiers. L'idée est donc venue de faire un lanceur amélioré, avec pourquoi pas une animation 3D d'un CPC ? Merci encore à *Make3DFrames* qui a permis de générer cette animation.

Puis une référence au groupe Impact : le Sonic avec le logo Impact fut ajouté. Tout d'abord en tant qu'image fixe, puis avec une petite animation de la main du Sonic.

En attendant l'échéance de la BND Coding Party, je trouvais encore des optimisations de vitesse de code et de taille... Il restait un peu de place libre en mémoire. L'occasion pour ajouter une « cheat part »,

elle aussi avec un objet, pas 3D, mais à base de triangles. Encore quelques optimisations plus tard, des logos furent ajoutés, et la démo terminée pour fin octobre.

Reste-t-il encore des choses à faire sur CPC à base de triangles ? Je pense que oui. Peut-être faire une refonte complète de la routine de tracé de triangle pour gagner encore un peu en vitesse, et pouvoir animer des objets à la demi-frame (1 / 25 de seconde) ? L'adapter en mode 0 pour permettre d'avoir plus de couleurs dans les objets ? Optimiser la routine d'effacement en utilisant la pile ? Affaire à suivre... ■



ALMOST A CENT P CENT

#51 SOON...



Stan W DECKER



PASCAL VIELHESCAZE

« Février 1951, profession détective privé. Le froid figeait Paris et mes affaires, lorsque, une lettre, un appel, les souvenirs d'une enfance encore proche. Que de jeux dans les pièces délabrées du *Manoir de Mortevielle*... ». C'est, à quelques détails près, en ces termes familiers pour le fêru d'aventures numériques de la première heure qu'*Invitation* aurait pu s'ouvrir sur Amstrad CPC.

PAR TITAN

Partageant, en apparence, sensiblement la même philosophie, *Invitation* pourrait même aisément passer pour une coupable imitation du célèbre hit de Lankhor s'il n'avait pas, dans les faits, été édité un an auparavant ! Alors, le *Manoir de Mortevielle* aurait-il en réalité honteusement plagié *Invitation* ? Afin de répondre à ce mystère et faire la lumière sur cette intrigue, nous sommes allés à la rencontre de **Pascal Vielhescaze**, auteur d'*Invitation*, paru chez Loriciels en 1987.



UNE ÉTRANGE SOLLICITATION...

Il faisait particulièrement doux en ce soir de novembre 1950. Plongé dans la semi-pénombre de mon bureau, le regard rivé sur les touches émoussées de ma fidèle Remington, je rédigeais un énième article sur les phénomènes occultes et leurs mystères, une discipline dont je m'étais fait l'expert dans les milieux littéraires. Pourtant, ce soir-là, j'éprouvais beaucoup de mal à me concentrer. Mon esprit était ailleurs, irrémédiablement préoccupé par la présence sur mon bureau de cette énigmatique missive reçue le matin même. À l'intérieur de la lettre j'avais découvert une étrange invitation dont les mots n'avaient cessé de hanter mes pensées tout au long de la journée. Qui était donc ce mystérieux Baron dont je n'arrivais pas à déchiffrer la signature et qui me conviait à 18 heures précises dans son manoir ? En quel honneur cette réception était-elle donnée ? Pourquoi me portait-on autant d'intérêt ? Devais-je honorer ou bien décliner la proposition ? Prises dans un tourbillon infernal, les questions se bousculaient et s'entrechoquaient dans ma tête, si bien qu'afin de les faire taire je décidais d'aller trouver les réponses sur place. Perdu au beau milieu d'une forêt inquiétante et immergé dans une atmosphère fantomatique amplifiée par une brume

opaque omniprésente, le manoir, tel une montagne lugubre, se dressait majestueusement devant moi. Timidement éclairé par la lueur pâle et vacillante d'une lune blafarde, l'imposante bâtisse projetait une ombre inquiétante sur les terres alentours que seule la lumière aveuglante provenant de sa porte d'entrée laissée grande ouverte venait entamer. Malgré la sensation de malaise qui m'envahissait inexorablement, poussé par la curiosité, j'en franchissais le seuil. Au moment où la porte se refermait derrière moi dans un claquement sec et brutal, je réalisais que j'étais seul... Enfin, c'est ce que je croyais...



Ce manoir ressemble étrangement à un autre

Introduit de façon très énigmatique et plutôt ésotérique, c'est sous les traits d'un RPG virtuel qu'*Invitation* propose une aventure prenante dans laquelle le joueur aura pour mission de trouver une liste d'objets précis, de résoudre des énigmes et de combattre des créatures surnaturelles. Plongeant l'aventurier dans une atmosphère inquiétante et oppressante, le scénario d'*Invitation* est ainsi prétexte à une expérience ludique très particulière sur fond d'enquête et de sciences occultes, et dont le but ultime est de percer le terrifiant secret du manoir.

ENTRETIEN AVEC UN VENT PIRE

Cher Pascal, peux-tu nous retracer le cheminement qui t'a conduit jusqu'aux jeux vidéo ?

Je suis issu d'une génération qui a vu émerger les jeux vidéo puis les ordinateurs dans son quotidien, en partant du néant jusqu'à ce que cela fasse partie intégrante de la culture. Comme nombre d'enfants à cette époque, très tôt j'ai branché un *Pong* sur la télévision familiale. Je me souviens que la machine s'appelait OC 5000. On a vraiment du mal à imaginer aujourd'hui l'émerveillement procuré par le réglage méticuleux de l'image neigeuse d'un canal afin d'obtenir d'un seul coup une belle barre blanche, deux raquettes et deux scores bien nets sur fond noir ! Je devais avoir autour de 13/14 ans lorsque j'ai découvert ensuite la version tabletop de

Space Invaders dans le café situé en face de mon lycée. Si je peux aujourd'hui affirmer avoir passé pas mal de temps (et de pièces de monnaie) sur ce shoot antédiluvien, ce n'est finalement rien comparé au choc produit ensuite par *Pac-Man*. Je me souviens encore de la toute première fois où j'ai posé mes doigts sur la borne et je reste fasciné par la richesse des émotions que j'ai pu vivre sur ce jeu dont toute l'essence s'enclave sur un seul et unique écran !

Au même moment, les premières calculatrices programmables arrivaient sur le marché, me conduisant à faire rapidement l'acquisition d'une TI57 puis d'une TI58C, deux machines sur lesquelles j'effectuais mes premiers pas dans la programmation grâce, notamment, aux listings proposés dans le magazine *Jeux et Stratégie*. C'est d'ailleurs ce périodique qui a commencé à parler régulièrement des ordinateurs et des premiers jeux disponibles. J'étais en admiration devant les premiers screenshots de ses articles et je crois qu'en stimulant de la sorte mon imaginaire, les journalistes m'ont poussé à économiser pour acheter un ordinateur. Il faut dire que j'ai toujours été joueur : je joue au poker, aux échecs, au backgammon, au go, et à l'époque je jouais aussi beaucoup aux jeux de stratégie, aux wargames, ainsi qu'aux jeux de rôle papier. Avoir la possibilité de jouer sans limite a rapidement rendu l'ordinateur totalement évident et indispensable à mes yeux.

Cependant, à cette époque reculée de l'informatique, il y n'avait vraiment que 2 alternatives sérieuses : le TRS80 et l'Apple II, deux plateformes proposant les ludothèques les plus riches et variées. Le premier s'avérant beaucoup plus abordable que le second, mon choix s'est alors porté sur le VideoGenie System, un compatible TRS80 fabriqué à Hong-Kong. Je me souviens qu'avant même avoir reçu la machine, mes amis m'avaient acheté le jeu *Robot Attack* pour mon anniversaire. Lorsque j'ai



OC 5000

ouvert le paquet et qu'ils ont découvert une cassette audio incrustée dans une plaque de polystyrène, ils m'ont clairement avoué qu'ils n'avaient absolument aucune idée de ce qu'ils m'offraient !

L'Apple II était la Rolls-Royce des micros et dès que j'en ai eu l'occasion, j'ai investi dans un modèle, un écran et un lecteur de disquette. Je me souviens de la facture salée : plus de 16000 francs (environ 2450 euros), ce qui constituait une véritable folie à 16 ans. Mes deux premiers jeux m'avaient été offerts par mon frère et un ami. Il s'agissait de *Cannonball Blitz* (très inspiré de *Donkey Kong*) et *Wizard and the princess*, l'un des tout premiers jeux d'aventure graphique à avoir émergé de la nuée d'aventures textuelles qui sévissait alors.



Masquerade (Apple II)

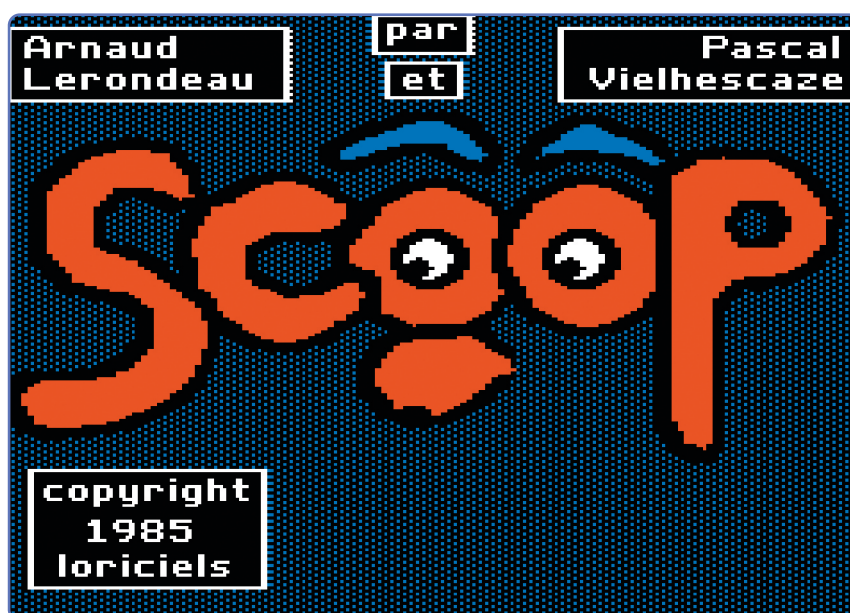
Nous étions deux amis du lycée à posséder un tel micro et nous passions beaucoup de temps ensemble à programmer, échanger des disquettes et s'enrichir mutuellement de tout cet univers. Nous jouions beaucoup et étions assez fans de jeux d'aventure. C'est par l'intermédiaire de *Masquerade*, un jeu américain qui nous avait tout particulièrement marqué, que nous avons décidé de créer nous aussi un jeu d'aventure. Je me souviens que sur le chemin du retour du lycée nous avons acheté une feuille géante de papier Canson, que nous avons punaisée sur le mur afin de structurer le scénario de notre futur jeu. Ainsi après lui avoir consacré toutes nos vacances de Pâques et d'été, *Scoop* est né, avec une innovation majeure pour l'époque puisqu'il s'agissait du tout premier jeu d'aventure capable de gérer la souris !

Raconte-nous l'histoire de la genèse d'*Invitation*.

Plusieurs années après *Scoop*, Christian, un ami qui possédait un Amstrad 464, a commencé à échanger avec moi sur l'idée de créer un jeu pour sa machine. À l'époque, je faisais des études de marketing et communication, et même si je jouais beaucoup (j'avais d'ailleurs l'un des premiers Macintosh, un Mac 1024k), créer un nouveau jeu n'était pas dans mes priorités. Cependant, l'idée de repartir pour une aventure de création m'a très vite séduit et lorsque mon ami a finalement souhaité ne pas aller plus avant, je m'étais déjà, pour ma part, totalement impliqué dans le projet.

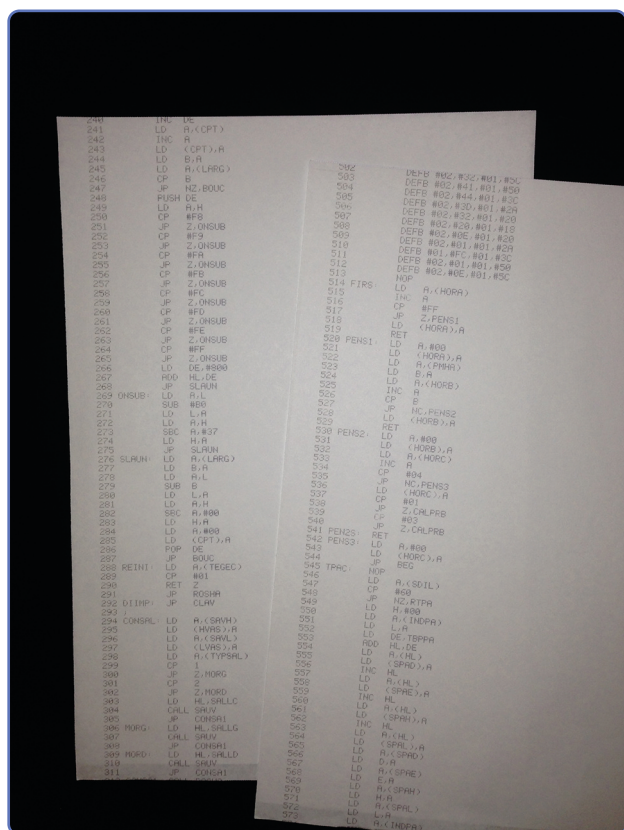
Évidemment, les temps avaient changé : j'avais 21 ans, j'étais étudiant, je sortais tous les soirs et je passais du temps avec ma petite amie. Le rythme de conception et de développement allait donc être radicalement différent. Alors pour avancer sur le projet, je me réservais tout simplement les nuits, ce qui n'était finalement pas un gros sacrifice étant donné que j'ai toujours eu besoin de peu de sommeil, même encore maintenant.

Il y a 3 catégories de jeux qui m'ont toujours attiré : les jeux de plates-formes, les jeux d'aventure et les RPG. À cette époque, j'avais déjà conçu un jeu d'aventure, et programmer un jeu de plates-formes ne me motivait pas plus que cela. De plus, je pratiquais beaucoup *Call of Cthulhu* et *Star Wars*, deux jeux de rôle très populaires à cette période. Aussi l'idée de développer un RPG s'est imposée de façon très naturelle. Effet de bord intéressant, le fait de réserver les heures les plus tardives de la nuit pour le projet, m'immergeait dans un état d'esprit parfaitement adapté pour un jeu de rôle orienté horreur.



Scoop (Apple II)

J'ai ainsi commencé par développer en assembleur (le langage machine) tous les outils dont j'avais besoin, à commencer par un compresseur et un décompresseur graphiques, ce qui m'a très rapidement permis de maîtriser les arcanes de l'Amstrad, les emplacements mémoire et tout ce qui était nécessaire pour mener à bien un développement performant et optimisé. Étant donné qu'il avait déjà édité *Scoop* sur Apple II presque quatre ans plus tôt, j'ai entre temps informé Laurant Weill (co-fondateur de Loricels) de ce que j'initiais. Cela m'a permis de disposer de matériel complémentaire, à savoir un Amstrad CPC 664 et une imprimante, pour organiser une véritable war-room *Invitation* permanente chez moi.



Le listing original d'*Invitation*

La trame du jeu était plutôt simple. S'appuyant essentiellement sur la découverte du manoir, *Invitation* exploiterait un système de gestion des objets et des combats afin de proposer au joueur un objectif unique mais aux exigences multiples, à savoir découvrir le secret des lieux sans périr ou devenir fou, mais surtout, avant que minuit ne sonne ! Aussi, ma première tâche a été de concevoir le manoir dans ses moindres détails afin de non seulement créer l'univers dans lequel l'aventure allait se dérouler, mais aussi de mieux m'immerger moi-même dans cet environnement, d'en connaître les moindres recoins et ainsi de plonger littéralement dans le jeu que je concevais simultanément. Travailler sur

l'architecture de la bâtisse m'a également poussé à créer l'interface globale du jeu telle qu'on la connaît aujourd'hui : avec des éléments principaux tels que les jauges de santé physique et mentale, l'horloge qui rythme la partie et apporte une pression supplémentaire, et enfin les zones d'affichage des informations, des objets et des rencontres inopinées.

Concevoir *Invitation* n'était qu'une de mes occupations parmi tant d'autres. Aussi, contrairement à ce qui s'était passé pour *Scoop*, le développement d'*Invitation*, plus sporadique, s'est étalé sur plusieurs mois. C'était un projet sur lequel j'avancais à mon rythme et dont je goutais chaque phase d'élaboration et chaque avancée lorsque je me remettait à l'ouvrage. Seule la finalisation a été très intense puisque, confronté à des problèmes de place mémoire, il m'a fallu avancer très rapidement et sans rupture afin de ne pas perdre la dynamique et la logique de cette ultime phase qui consolidait tous les éléments du jeu. Sans mémoire suffisante pour charger l'assembleur, j'ai même été contraint de coder certaines zones d'image directement en hexadécimal, ce qui m'a tenu éveillé plus de 72 heures d'affilée afin de ne pas perdre le fil. Je crois qu'à cet instant, mon cerveau n'était plus que le miroir intégral de la carte mémoire de l'Amstrad !

Ensuite, j'ai immédiatement enchaîné sur l'étape suivante puisque le matin même j'ai relancé et revérifié tous les tests que j'avais mis en place pour valider le bon fonctionnement du jeu. J'ai alors compilé le programme en plusieurs versions sur disquettes et j'ai pris le volant en direction de Rueil-Malmaison pour remettre les masters à Laurant Weill directement chez Loricels.

Mise à part une courte période durant laquelle je me suis rendu disponible pour participer au lancement du jeu, en écrivant par exemple le manuel d'utilisateur ainsi que le texte au dos de la jaquette, ou encore en répondant à quelques interviews pour la presse, ce fut la fin de l'aventure pour moi. J'ai vraiment le souvenir d'avoir littéralement instantanément tourné la page et basculé sur d'autres projets sans même regarder une seule fois en arrière. Je me souviens même avoir été presque surpris de retrouver *Invitation* un peu plus tard sur les présentoirs de la Fnac Wagram : j'affichais alors un détachement assez surprenant si je repense à l'énergie que j'avais mise dans cette création.

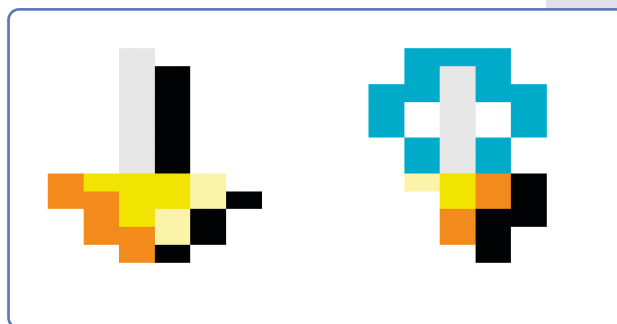
Comment as-tu imaginé ce scénario et quelles ont été tes éventuelles sources d'inspiration ?

Ma principale source d'inspiration pour *Invitation* a bien évidemment été puisée dans l'univers fascinant d'H.P. Lovecraft, et plus particulièrement dans celui

du jeu de rôle *Call of Cthulhu* auquel je jouais beaucoup. Lovecraft a créé une mythologie complète à travers ses romans et nouvelles. On retrouve une richesse inégalée dans les descriptifs d'environnement, dans le bestiaire et dans la manière avec laquelle il détaille les sentiments d'horreur. Une conséquence majeure de cette référence est la présence des deux jauges sur l'interface : une jauge de santé physique et une jauge de santé mentale. On peut ainsi perdre dans *Invitation* en devenant fou, sans mourir pour autant.

Tu as partagé l'expérience avec Philippe Gueneau de Mussy. Quel a été son rôle dans le développement ?

Philippe était un ami graphiste qui s'intéressait aux possibilités des ordinateurs dans ce domaine. Je lui ai proposé de dessiner quelques monstres pour se faire la main et bien évidemment cela m'a un peu soulagé sur le planning, mais c'était surtout l'occasion de partager quelques moments avec lui et d'avoir un petit peu de recul sur certains aspects du jeu. C'est par ailleurs Philippe qui a réalisé l'illustration de la jaquette d'*Invitation*.



mode zoom et de m'être attaché avant tout à la perception du rendu à l'écran. Mes choix de palette de couleurs pour chaque lieu privilégiaient des teintes proches pour générer des effets d'ombre. Par exemple, les bougeoirs et les appliques jouent vraiment sur cette approche : un pixel jaune vif adossé à un pixel orange rendait un effet doré très classe. J'ai un très bon souvenir de navigation libre dans le manoir sans que le jeu ne soit activé. En effet, durant l'élaboration du jeu, je ne me lassais pas de me promener dans ce lieu que j'avais créé : j'adorais évoluer de pièce en pièce en scrutant les détails des meubles et de la décoration.



Aussi esthétique soit-il, le graphisme du manoir d'*Invitation* révèle des intérieurs impeccables et immaculés où toute chose est à sa place, où chaque meuble est bien rangé, et où le moindre grain de poussière n'a pas sa place. Ne pensais-tu pas que cette perfection visuelle fasse naître un étrange paradoxe pour un château abandonné et infesté de monstres ?

Au début de l'aventure, lorsque l'on arrive au manoir, celui-ci n'est pas censé être abandonné puisque l'on est invité par son

Mis à part quelques graphismes secondaires, tu es à l'origine de tout le design du jeu. Peux-tu nous parler de cette phase ?

Invitation a été conçu pour que chaque nouvelle aventure soit une expérience unique. Ainsi à chaque partie, les objets clés, les livres et les monstres voient leurs emplacements totalement redistribués. Pour cette raison, je voulais que le manoir devienne un endroit familier afin que le joueur puisse retrouver facilement, à chaque partie, ses lieux ou ses pièces favorites. J'ai donc élaboré avec soins les plans du manoir et la personnalisation des pièces qui le composent. Concernant les décors et les objets, j'ai surtout le souvenir d'avoir travaillé pixel par pixel en

propriétaire et que l'on s'y rend sans méfiance ! En fait, mon idée initiale était de faire évoluer l'environnement et l'atmosphère du manoir au fur et à mesure de la soirée. Ainsi si l'on arrivait dans un manoir très clean et très luxueux en début d'aventure, le lieu devait ensuite graduellement se modifier pour devenir de plus en plus étrange et oppressant, notamment lorsque l'on perdait de la santé mentale. C'est amusant, je réalise maintenant que c'est exactement ce qui a été magistralement fait et de façon beaucoup plus ambitieuse bien plus tard avec *Eternal Darkness* sur Nintendo Gamecube. Mais au final, je n'avais plus la moindre place mémoire pour aboutir à ce résultat et le manoir est donc resté effectivement bien propre et bien rangé. Je me

souviens d'une critique dans un magazine qui avait mis l'accent sur cet aspect du jeu et j'étais réellement mortifié de lire ça, en plus d'être frustré de ne pas avoir pu faire ce que j'avais prévu.

Invitation offre la possibilité d'observer les pièces selon plusieurs angles de vue. Comment t'est venue cette idée ?

La décision de concevoir ce système de vues multiples a été avant tout guidée par le confort de navigation pour l'utilisateur. Encore une fois, je souhaitais que chacun ait rapidement ses propres repères au sein du manoir et cela m'a semblé être la solution la plus élégante. En termes de programmation, cela permettait aussi de classer les vues de façon très logique et très mathématique, ce qui me simplifiait pas mal la conception, particulièrement lorsque je reprenais certaines zones de développement après avoir travaillé sur d'autres éléments du jeu en parallèle. Enfin, c'était justement une solution peu gourmande en mémoire puisque le nombre de vues différentes était prédéfini et que je pouvais suivre une même architecture de programmation en jouant simplement sur les couleurs et l'emplacement des objets et du mobilier, sans réel souci de perspective ou de proportions. Et puis j'avais surtout conçu tout spécialement pour le jeu un compresseur et un décompresseur graphiques qui me permettaient d'optimiser considérablement le poids mémoire des images. Au final, cela me donnait la possibilité de produire des combinaisons d'environnements largement supérieures à mes besoins réels pour le jeu.

À une époque où la grande majorité des jeux d'aventure ont recours à de fastidieux et rigides analyseurs syntaxiques, *Invitation* avance quant à lui un système convivial et extrêmement simple d'interaction entre l'homme et la machine se limitant à seulement 5 touches sur le clavier. Peux-tu nous en dire plus sur ce concept révolutionnaire ?

Je ne sais pas s'il faut qualifier cette approche de révolutionnaire, mais il s'agissait avant tout de privilégier le plaisir du jeu et la perception de l'atmosphère portée par l'aventure. Naviguer avec des gestes simples en n'utilisant qu'une touche pour chaque commande permettait de focaliser son attention sur l'environnement. En y repensant, j' imagine que c'était aussi lié à mon expérience de *Scoop* sur Apple II puisque ce jeu était le premier jeu d'aventure utilisable avec une souris. L'interface me semblait tellement plus simple que j'ai instinctivement construit celle-ci de la même façon... mais sans souris !

Comme dans toute énigme qui se respecte, l'aventure invite parfois le joueur à décoder certains textes provenant des livres trouvés dans les bibliothèques du manoir. Pour prendre connaissance de la clef de décryptage il faut alors avoir recours au manuel vendu avec le jeu. En plus d'accentuer l'immersion dans l'aventure, était-ce un moyen de limiter le piratage ?

Scoop a été certainement l'un des jeux les plus piratés de son époque. Aussi, je ne cache pas que j'avais effectivement cela en tête en imaginant le procédé de décryptage intégré dans *Invitation*. Ceci dit, les bibliothèques et les textes cryptés que l'on y trouve sont un élément fondamental du jeu et de son atmosphère. Cela fait partie du corpus de *Call of Cthulhu* et il aurait été impensable pour moi de ne pas intégrer cette dimension dans *Invitation*.



Invitation

Renforçant la sensation de solitude, aucun son ne vient troubler l'aventure. Était-ce un choix délibéré ou bien une conséquence induite par l'absence de musicien ?

En y repensant maintenant, c'est à mon sens une erreur de ne pas avoir intégré ne serait-ce qu'une mélodie répétitive un peu angoissante. Pour être très honnête, je crois que je ne me suis même pas posé la question à l'époque. Concevoir le jeu, développer le programme et élaborer les graphismes me semblaient couvrir toute la check-list ! J'ai tout simplement totalement occulté cet aspect bruitages et musique dans *Invitation* !

Y a-t-il des choses que tu avais prévues d'intégrer mais que tu as finalement été contraint d'abandonner ?

En plus de l'évolution de l'environnement dont je parle plus haut, j'avais également prévu un jeu beaucoup plus riche dans la diversité des monstres et des situations de combat ou d'exploration. Mais je n'ai pas été aussi loin dans la démarche. Et pour une fois, ce n'était pas forcément lié à un manque de

mémoire. En effet, il est toujours possible de repenser la structure de programmation pour optimiser et gagner de l'espace. Non, je pense plutôt que c'était dû au fait que j'étais seul pour gérer et couvrir simultanément tous les aspects du développement, ce qui ne permettait pas d'être lucide à tous les niveaux. D'une certaine façon, l'ouvrage était tellement titanesque que j'ai certainement assouvi mon besoin de création plus rapidement qu'à plusieurs, même si j'avais sans cesse de nouvelles idées à implémenter. Enfin, il manque certainement tout ce que des échanges et des débats entre amis auraient pu apporter pour enrichir une expérience et repousser sans cesse les limites.

Peux-tu nous donner des détails sur la phase de conception du jeu ?

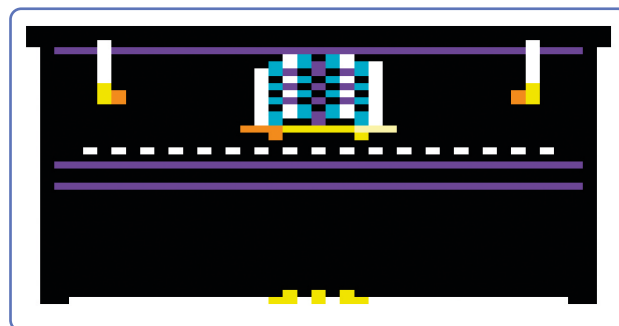
Avant même d'amorcer la conception du jeu, je m'étais en quelque sorte entraîné sur la configuration et les contraintes de l'Amstrad en développant le fameux compresseur/décompresseur graphique. Je me souviens même avoir poussé le vice jusqu'à traiter deux modes de compression distincts afin de ne garder à chaque fois que le mode le plus léger pour chaque image ou chaque objet. L'intégration d'un petit flag (un repère) au début de chaque graphisme permettait alors au programme de savoir quel mode devait être employé pour la décompression.

J'ai également dû concevoir un générateur graphique de caractères puisque celui-ci n'était pas disponible dans le système. En y repensant, la mise en place de l'horloge et la gestion du temps a été aussi assez pointue. Afin que le temps s'égraine de façon régulière, le programme était conçu dans son intégralité comme une boucle permanente afin que les phases temporelles soient régulières et homogènes. Chaque interruption liée à l'utilisateur pondérait cette gestion du temps afin de la rendre la plus précise possible.

Enfin, j'ai écrit et testé de nombreuses formules statistiques pour que le jeu soit le mieux équilibré possible entre la progression, les combats et leur difficulté.

Quant à la partie graphique, ne disposant pas de tablette (je ne sais même pas s'il en existait à l'époque pour Amstrad), j'ai tout dessiné au joystick en mode zoom quasi permanent. Dans une résolution aussi basse que le mode 0 (160 × 200 pixels), il est réellement impressionnant de constater à quel point un pixel judicieusement positionné peut totalement modifier le rendu final. Je passais donc un temps incroyable à tester à la loupe une couleur ou un emplacement de pixel, pour ensuite revenir en mode d'affichage normal et découvrir ce que l'œil perce-

vait au final. À titre d'exemple, je suis assez fier du piano laqué noir et en l'observant attentivement, on se rend compte que la perception d'ensemble se joue vraiment à quelques pixels bien placés !



Sais-tu quel accueil a reçu *Invitation* auprès des joueurs et de la presse lors de sa sortie ?

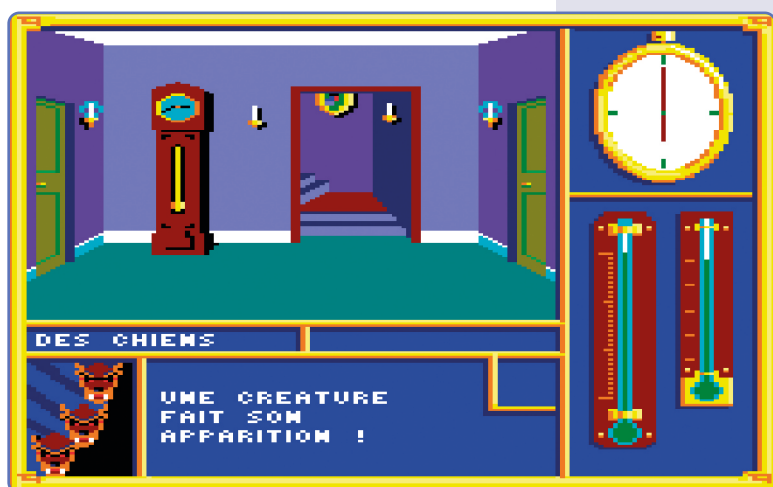
Les critiques et les articles étaient plutôt positifs. De toute façon, je n'avais pas d'objectifs particuliers si ce n'est, encore une fois, ressentir le plaisir de concevoir un jeu et de le voir prendre vie. Donc pour moi, l'objectif principal était atteint ! J'imagine par contre que Loricels avait, comme tout éditeur, un objectif de vente bien défini. Je n'ai pas de trace des chiffres mais je pense qu'*Invitation* n'a pas été ce que l'on qualifierait de best-seller. Ceci dit, puisque j'avais conçu le jeu en freelance, l'investissement de Loricels n'était que commercial et marketing. Je pense qu'éditer plusieurs jeux en espérant que les ventes de l'un d'eux soient stratosphériques était une forme de stratégie commerciale. J'ajouterais qu'il n'y a eu absolument aucune intervention de leur part sur le jeu lui-même et qu'avec le recul, il me semble qu'ils ont parfaitement joué leur rôle d'éditeur, que ce soit en distribution ou en relations presse.

Malgré de très bonnes critiques, certains magazines (*Tilt* pour ne pas le citer) ont toutefois reproché à *Invitation* de ne pas être assez dynamique. La faute, d'après les pigistes, à une gestion et à un affichage des monstres en encart qui procure un côté un peu trop figé à l'action. Que penses-tu de cette critique ?

C'est vraiment intéressant parce que cette critique est en fait totalement assumée de mon côté. Pour moi, *Invitation* a toujours été une transcription assez fidèle d'un RPG papier en jeu vidéo. J'ai passé des centaines d'heures des années plus tôt sur *Wizardry* sur Apple II qui était totalement minimaliste, évidemment du fait de la technologie de l'époque et surtout parce que c'était l'un des tout premiers RPG sur ordinateur. *Invitation* devait pour moi laisser une grande place à l'imagination. Et à mes yeux, ces limites graphiques et technologiques de l'époque

avaient pour conséquence que les jeux étaient justement plus spectaculaires et plus prenants lorsqu'ils ne cherchaient pas à être très figuratifs. C'est un peu similaire à la perception que l'on a de la qualité des premiers jeux 3D, finalement moins beaux et moins séduisants que beaucoup de jeux 2D de la même période, en tout cas, je l'ai toujours ressenti ainsi.

Ceci dit, je pense rétrospectivement que j'aurais pu afficher plus d'informations lors des combats, notamment en informant de l'état des monstres au fur et à mesure, peut-être avec des éléments graphiques un tant soit peu animés, ce qui aurait effectivement dynamisé un peu plus ces phases particulières.



Invitation

Qu'est-ce qui t'a le plus amusé dans la création d'*Invitation* et, inversement, qu'est-ce qui t'a semblé le plus fastidieux ?

L'expérience de la conception et de la réalisation de *Scoop* sur Apple II plusieurs années auparavant avait été un plaisir commun avec mon ami Arnaud et j'en ai un souvenir absolument magique d'excitation, de partage et d'échanges permanents. Ce fut vraiment une aventure unique.

Paradoxalement, ce qui m'a le plus motivé dans *Invitation* a été d'être totalement seul face au défi que cela représentait. *Invitation* devenait un projet presque introspectif pour moi et je sais que j'assouvissais de nombreuses choses très personnelles à travers cette création. Mais avant tout, c'était l'opportunité de maîtriser un jeu vidéo dans toutes ses dimensions. Il n'y a d'ailleurs rien eu de contraignant dans cette période qui était vraiment plus une expérience lu-

dique pour moi que quoi que ce soit de pénible ou de fastidieux. Et si jamais je saturais un peu sur la programmation ou si je bloquais temporairement sur une difficulté technique, il me suffisait de passer à la conception d'éléments graphiques, et inversement. C'est évidemment le bon côté d'être en charge de l'intégralité du projet !

Pour quelle raison *Invitation* n'a-t-il pas connu d'adaptation vers les machines 8 bits concurrentes de l'Amstrad CPC ?

Mon plaisir était simplement de concevoir et développer un jeu. L'adapter moi-même sur une autre plate-forme aurait été quelque chose d'extrêmement fastidieux et c'était absolument inconcevable pour moi. C'est réellement l'acte de création qui me pousse et cela ne supporte pas la répétition. Cependant, l'éditeur aurait effectivement pu en décider tout autrement, mais le jeu n'ayant finalement pas été un hit, j'imagine que cela n'avait pas de raison d'être pour eux.

Qu'éprouves-tu aujourd'hui, plus de 25 ans après, vis-à-vis d'*Invitation* ?

J'ai conservé des packs de cassettes et disquettes du jeu, ainsi qu'un press-book assez complet renfermant les articles et interviews parus à l'époque. Aussi, quand chez moi je retombe sur ce grimoire par hasard, c'est tout simplement un morceau de mon adolescence que je peux tenir entre mes mains, un peu à la manière d'une madeleine de Proust ! Quant au jeu lui-même, il est toujours très difficile pour moi d'en être totalement satisfait car j'ai le réflexe de raisonner par rapport à ce que je connais maintenant et par rapport à mon expérience... Et je vois à présent très clairement ce qui en aurait fait un jeu bien meilleur. Mais je crois qu'il faut vraiment le prendre tel quel, comme un témoignage, un reflet de ce qui se faisait à l'époque. Mais surtout, je le ressens comme une trace concrète et physique d'une période de ma vie et, comme cette interview le prouve, c'est un vrai catalyseur de souvenirs pour moi, ce qui est plutôt agréable ! ■





64 NOPs

Rédaction : Hicks, toms

Ont contribué à ce numéro : Demoniak, Eliot, Golem13,
Hwikaa, OffseT, Pascal Vielhescaze, Titan, Zik

Couverture : Made (2D.FR)

Mise en page : toms, Hwikaa

Pubs : Barjack, Hwikaa, DarkSteph, Oncle Ced

Merci aux auteurs des photos publiées dans les articles !

Première édition · 110 exemplaires

Pour contacter la rédaction :

Email : admin@memoryfull.net

Sites web : 64nops.wordpress.com · www.memoryfull.net

CPC-ANACHRONIE

NIVEAU 1

LE MONDE AMSTRAD

- Test de matériels
- Les SUCRES 2022
Jeux Amstrad récompensés par la rédaction
- Un chien nommé Moritz
- Jeux 2023 CPC et GX4000

POP CULTURE GEEK

- Dossier Pokémon
- Rock On avec Alicia F !
- Philippe Ulrich
« L'Arche du Cœur Blood »
- Entretien avec Lord Paddle

200% d'anachronisme dans ta vie !

Novembre 2023 - SEMESTRIEL - 15€

CPC-Anachronie, c'est ton nouveau magazine semestriel de 108 pages consacré au monde Amstrad et à la pop culture geek !

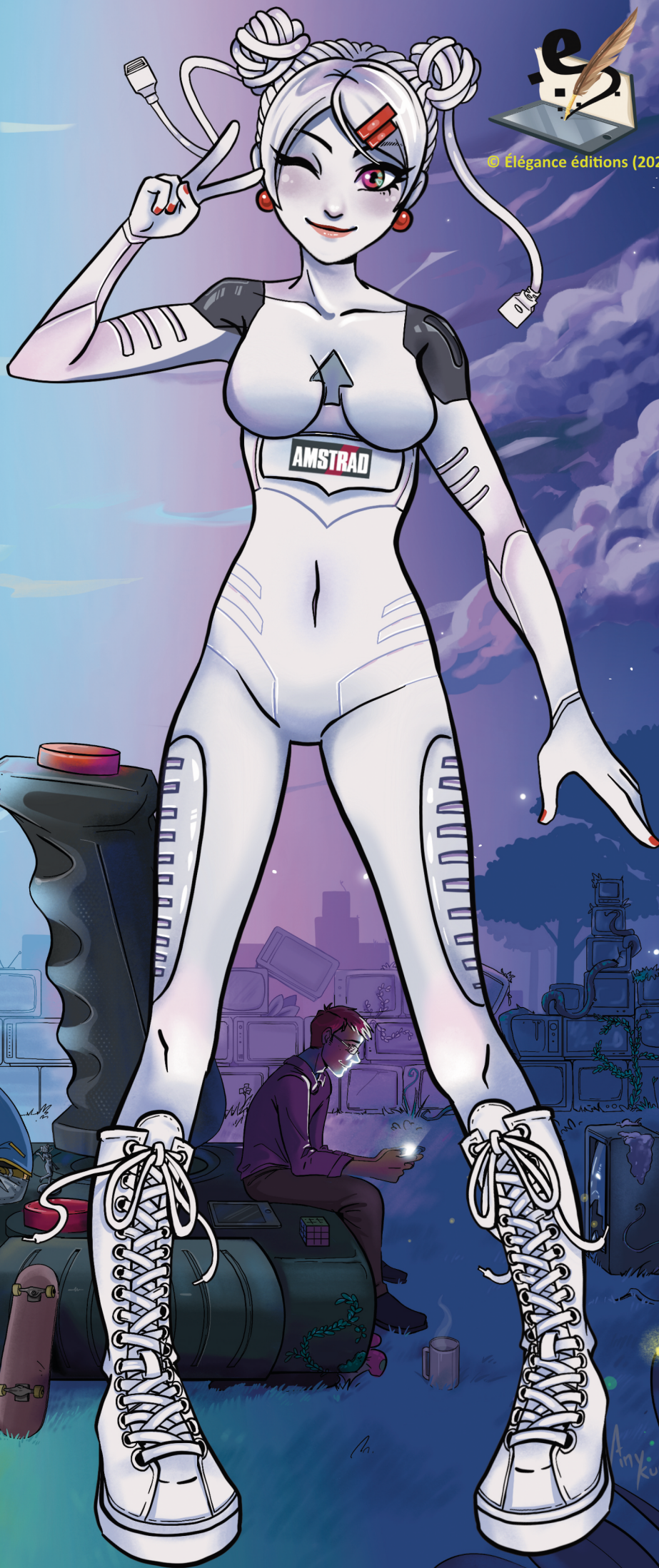
CPC-Anachronie, c'est 200% d'anachronisme dans ta vie rien que pour toi !

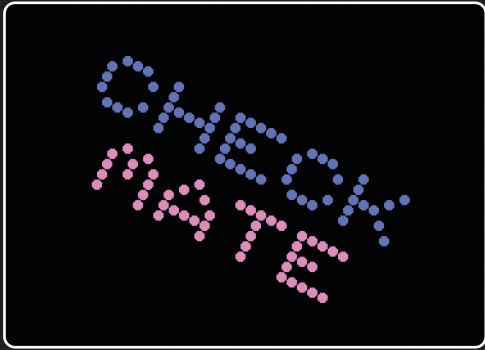
Disponible sur :

<https://elegance-editions.eproshopping.fr/>

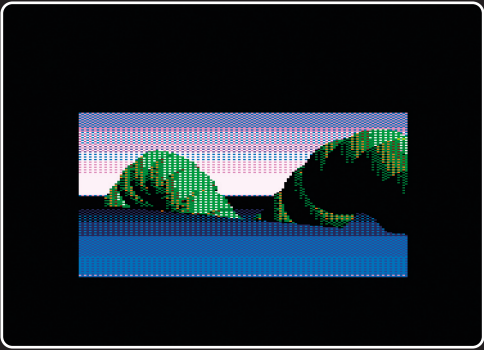


© Elegance éditions (2023)

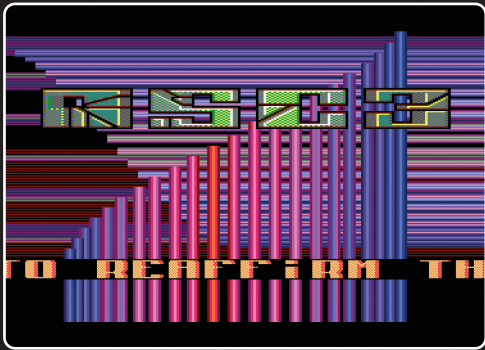




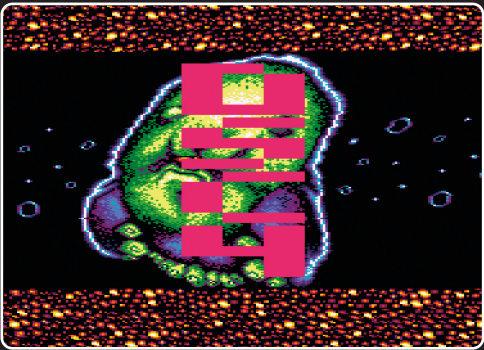
Checkmate (Pulpo Corrosivo)



Emotion Trouble (Ramdam Brouhaha)



Onescreen Colonies #3 (Vanity)



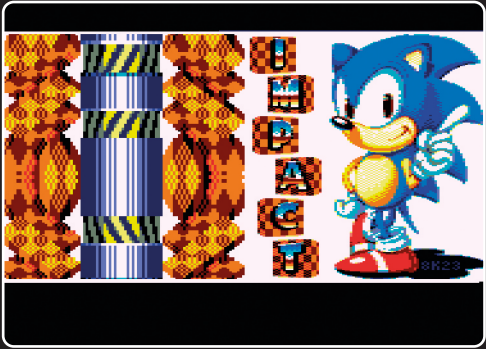
DSC #4 (Logon System)



Stand Up! (Benediction)



FatMag 2 (Praline)



Still The One (Impact)