Atari ST : Overscan

• Sengan Baring-Gould
Docteur en Intelligence Artificielle et CEO d'Ansemond LLC. Auteur du livre Sams Teach Yourself Cocoa Touch Programming in 24 Hours (Développement d'applications sur iPhone en 24h).

Overscan ? Fullscreen ? Hardscroll ? Ces mots ont été depuis longtemps réservés aux passionnés et codeurs sur Atari ST. Il est temps aujourd'hui de vous révéler comment les développeurs de l'époque dépassaient les capacités techniques et hardware de la machine.

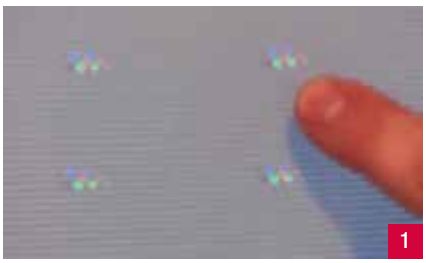
Petit retour en arrière

Il faut se souvenir qu'à l'époque - il y a donc 30 ans - l'Atari ST pouvait être branché soit à un téléviseur soit à un moniteur, tous deux utilisant un tube cathodique. Ces écrans pouvaient afficher 625, 525 ou 500 lignes au total, qui étaient balayées par un ou trois faisceaux d'électrons 50, 60 ou 71 fois par seconde respectivement. Afficher un pixel sur l'écran consiste à régler le voltage des faisceaux d'électrons quand ils ciblent ce pixel. Le système vidéo de l'Atari ST doit lire la mémoire dédiée aux graphismes à montrer sur le moniteur et régler le voltage qu'il envoie au moniteur au même moment que les faisceaux d'électrons ciblent chaque pixel de l'écran pour que l'image voulue apparaisse sur l'écran. [1]

Le système vidéo de l'Atari ST est composé de différents composants dont la principale puce dédiée est appelée le **SHIFTER**, qui est aidée par les composants dénommés la **GLUE** et la **MMU**. C'est la **MMU** qui accède à la mémoire à accès aléatoire (RAM) via le bus 16-32 bits du ST et envoie ce qu'elle lit au **SHIFTER**. Le **SHIFTER** utilise ces informations pour régler le voltage qu'il envoie au moniteur.

Parce que les téléviseurs et les moniteurs avaient des réglages différents, les côtés des images n'étaient souvent pas visibles. Pour éviter que les utilisateurs rouspètent, la plupart des ordinateurs de l'époque affichaient des bordures autour de l'image utile, c'est à dire la partie de l'écran sur laquelle on peut afficher des graphiques. Par exemple, un téléviseur Français aux normes SECAM avait 625 lignes entrelacées. C'est à dire que les faisceaux d'électrons balayaient alternativement 313 et 312 lignes de l'écran 50 fois par seconde.

L'Atari ST trichait et envoyait 313 lignes 50 fois par seconde. Un téléviseur Américain aux normes NTSC avait par contre 525 lignes entrelacées, et l'Atari ST lui envoyait 263 lignes 60 fois par seconde. Le moniteur noir et blanc d'Atari balayait environ 500 lignes environ 71 fois par seconde en mode MONOCHROME. En particulier, l'image utile apparaît dans 200 de ces lignes (basse résolution de 320x200 et moyenne résolution de 640x200). Les lignes restantes consistent en des signaux de synchronisation et surtout depuis la bordure du haut et celle du bas. Puisqu'il reste 63 lignes pour un téléviseur Américain, l'image utile commence à la 34e ligne affichée, et s'arrête à la 234e ligne. La bordure continue alors jusqu'à la 263e ligne. Par contre, pour un téléviseur Français, il reste 113 lignes, et l'image utile commence selon le modèle d'Atari ST, soit à la ligne 47, soit à la ligne 63. C'est la **GLUE** qui s'occupe de produire ces signaux de synchronisation, et qui décide quand le **SHIFTER** doit afficher une bordure et quand il doit afficher l'image utile.


1


2


3

Hypothèses

Puisque la fréquence du moniteur est programmable, on peut se demander si on peut « embrouiller » le système vidéo pour qu'il supprime les bordures, sans embrouiller le moniteur qui lui attend que les signaux venant du système vidéo soient cadencés de façon très précise pour que les pixels soient placés correctement sur l'écran. La réponse est affirmative. On peut, en passant à 60Hz (mode NTSC) à la fin de la ligne 33 et en revenant å 50Hz (mode SECAM) juste après le début de la ligne 34. Pour ce faire il suffit(!) de synchroniser son programme avec le faisceau d'électrons du moniteur. Heureusement, le compteur vidéo du système vidéo révèle l'adresse mémoire que le système vidéo lit pour envoyer les pixels de l'image utile au moniteur. Le compteur est donc synchronisé avec le faisceau d'électrons du moniteur, et peut servir pour synchroniser un programme tournant sur le 68000 avec ce faisceau d'électrons. Cela permet de changer la palette plusieurs fois par ligne pendant l'affichage de l'image (images Spectrum 512), et d'éliminer les bordures (mode Overscan). En exploitant la différence entre les différents modes vidéo, on parvient à éliminer toutes les bordures (haut, bas, droite et gauche) en changeant entre les trois modes vidéo (SECAM, NTSC, MONOCHROME) à des moments bien précis, et pour des durées bien déterminées, à chacune des 274 lignes de l'image utile sans bordures.

Voici les différents types de bordures entre un écran couleur en BASSE résolution à 60Hz [2] et un écran monochrome en HAUTE résolution à 50Hz [3].

Puisque les changements de mode vidéo requis pour générer un Overscan complet ont lieu à divers moments pendant 90% du temps machine disponible, tout programme qui fait quelque chose d'utile doit être parsemé de routines qui changent le mode vidéo à des moments bien précis. Pour créer ces programmes, il fallait coder en assembleur et prêter attention à la durée de chaque instruction. On pouvait alors diviser son programme en séquences d'instruc-

tions de la durée requise qu'on pouvait placer entre les routines changeant le mode vidéo. Un travail d'optimisation acharné donc..

Synchronisation et optimisation

Comme le système vidéo et le programme doivent être synchronisés, en général le code exécuté pendant l'Overscan n'a aucun branchement conditionnel: on veut éviter de calculer plusieurs durées par séquence d'instruction (à cause des chemins multiples à travers le code). Le code qui tourne pendant un Overscan sert donc surtout à faire des tâches répétitives, d'une façon assez similaire au code d'aujourd'hui qui est destiné à tourner sur les cartes graphiques. Il faut se rappeler qu'à l'époque, seulement les registres du système vidéo étaient documentés, mais pas leur fonctionnement interne. Tous les changements de mode vidéo ont été découverts en expérimentant avec des programmes tests. D'ailleurs, bien qu'on ait découvert les changements de mode vidéo nécessaires pour éliminer toutes les bordures, l'image obtenue n'était pas stable. Plus d'expérimentation révéla qu'un changement de résolution pour une durée courte à la fin de chaque ligne stabilisait l'image, mais on ne savait pas pourquoi ?!

La découverte de l'Overscan a permis des innovations graphiques techniques telles que le Hardscroll. Sur l'Atari STE, il suffit de changer l'adresse du début de la mémoire vidéo pour déplacer l'écran visible (multiples de 16 pixels) vers la gauche ou vers la droite, et de lignes entières vers le haut ou le bas. Mais ce n'était pas possible sur l'Atari ST. Du bon code assembleur prend environ 140000 cycles pour déplacer les graphiques d'un écran sans Overscan, et presque le double pour un écran en Overscan. 140000 cycles correspondent à 85% du temps disponible, puisqu'on ne dispose que de 160256 cycles si on veut faire des animations fluides qui tournent 50 fois par seconde.

Le Hardscroll a pour origine l'observation suivante: une ligne sans bordure consomme plus de mémoire vidéo qu'une ligne avec des bordures. Le compteur vidéo aura donc plus avancé à la fin d'une ligne sans bordure qu'une ligne avec. Il est donc possible de trouver des combinaisons de lignes avec ou sans bordures à droite ou à gauche qui permettent de spécifier l'adresse du début de la mémoire vidéo qui suit au mot près. Cela ne prend pas plus que 3000 cycles, soit presque 50 fois moins de temps : un gain immense permettant la réalisation de Scrollers en Overscan totalement fluides utilisés dans les jeux tel que **Lethal Xcess** d'*Eclipse Software Design* ou **Enchanted Lands** de *Thalion Software*

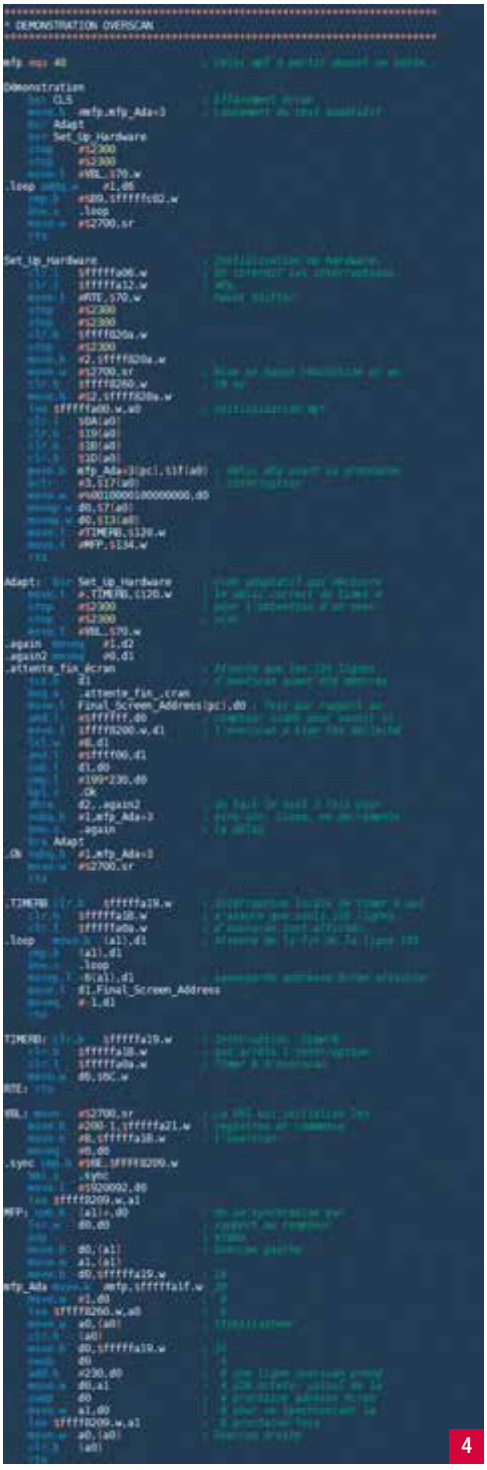
Pour la petite histoire

En 1989 je désirais passionnément faire mon propre Overscan. Après beaucoup d'expérimentations, j'ai réussi à faire un Overscan stable et un Hardscroll fluide sur Atari STF. Mais je voulais aussi comprendre comment fonctionnait le système vidéo de l'Atari, pourquoi l'Overscan était possible et pourquoi il fallait absolument le stabiliser. J'ai donc tenté de recréer ce système vidéo sur papier en utilisant tout ce que je savais du hardware, et en écrivant de nombreux programmes de test qui permettaient de décider entre toutes les hypothèses possibles. J'ai documenté les résultats de ces recherches dans mes articles publiés dans *ST Magazine* de l'époque. [4]

Un de ces résultats a été la découverte du Hardscroll 4-bits, une méthode pour décaler l'image utile de la droite vers la gauche sur un simple STF, ce qui était bien utile pour les Scrolling Horizontaux.

Codeur du groupe ST CONNEXION, j'ai codé en 1991 un écran démontrant les nouvelles capacités de l'Atari ST avec une musique Soundtrack pour la méga démo **Punish Your Machine** du groupe Delta Force intitulé *Let's Do The Twist Again*.

Quelques articles ont ensuite servi à créer des ré-implémentations de l'Atari ST sur des circuits intégrés reprogrammables FPGA et pour les émulateurs Atari ST. Et cette année, un hacker plus assidu que moi, *Jorge Cwik* a démonté complètement la puce du **SHIFTER** et il a trouvé les registres internes dont j'avais démontré l'existence pour obtenir les effets que j'observais dans mes programmes de test à l'époque. Pour les détails précis des valeurs nécessaires pour obtenir un Overscan je vous renvoie donc à ces articles que vous trouverez dans les ST Magazines numéros 51, 52, 55 et 70. Le site Internet http://abandonware-magazines.org/ de *Fréderic Letellier-Cohen* en a préservé des copies que vous pouvez consulter en ligne ou en téléchargement. Vous y trouverez aussi un historique des programmeurs qui ont découvert chaque aspect de l'Overscan et du Hardscroll. Finalement, vous y trouverez aussi un outil, l'intégrateur, qui permet de calculer comment décomposer votre programme pour y insérer les changements de mode vidéo nécessaires pour obtenir l'Overscan.


4

Depuis 1989, il y a eu quelques découvertes de plus sur le mode Overscan sur l'Atari ST via le monde de la Démoscene. *Troed Sångberg* (codeur du groupe SYNC) a résumé mon travail et ce qui a été découvert depuis sur son blog https://blog.troed.se/2015/12/23/overscan-and-sync-scrolling/.

•