Sprite Hardware 80's: Le grand comparatif

Les tableaux comparatifs de specs c’est rigolo mais c'est une chimère. On appréciait ces comparatifs dans les magazines des années 90 mais avec le recul il s’avère que les chiffres étaient souvent approximatifs voire faux, et même dans le cas contraire, ce type de comparatif n'est jamais très pertinent. Il est très périlleux de vouloir résumer des machines en quelques chiffres, chacune étant un patchwork complexe de multiples commodités et contraintes qui leurs sont propres. L’apothéose de tout ceci fût de résumer les machines à un nombre de bit, ce qui nous a donné la fameuse Jaguar 64bit!

Même en étant conscient de tout ça, j’ai eu une soudaine envie de m’essayer à cet exercice sous la forme d'un tableau que j'ai ensuite accompagné d'explications spécifiques à chaque machine pour les plus téméraires et curieux, dans la limite de mes connaissances. Je me suis focus sur un seul élément, les sprites hardwares, ça simplifie un peu la tâche. Mais sur un panel tout de même assez large et varié. Une petite vingtaine de machines avec beaucoup de consoles mais aussi du micro et de l'arcade.

Les chiffres du tableau expriment les performances hardware du chip graphique seul, sans aucune intervention du CPU. Il y a bien sûr quelques choix arbitraires dans ce tableau. Certaines machines proposent plusieurs modes vidéo et résolution d'où parfois des valeurs doubles dans les cases du tableau. Mais comme on parle de sprite et d'action, j'ai exclu certains mode vidéo "hires" qui ne sont pas adaptés ou utilisés in-game dans les jeux (par exemple le mode 640x sur GX4000, le mode 512x sur PCE, MSX2 et SNES). Un certain nombre de vieilles machines ont aussi des fonctions de "magnificence" qui permet de multiplier la taille d'un sprite par un facteur entier (à ne pas confondre avec un véritable effet de scaling) que je n'ai pas pris en considération quand cette magnificence consiste à dégrader la résolution du sprite par rapport à celle du background. C'est de toute façon un usage rare et anecdotique. Tout comme la possibilité qu'offrent certaines machines de pouvoir modifier la liste de sprite pendant le balayage, au raster, pour faire du multiplexage software et ajouter des sprites supplémentaires à l'écran. Ça ne fait pas vraiment partie du support natif hardware des sprites, ça nécessite l'intervention du CPU, en plus de ne pas être vraiment quantifiable. Mais vous trouverez tout de même toutes ces informations dans les chapitres dédiés à chaque machine, à défaut de les avoir dans le tableau.

Un peu de contexte et d'histoire

Comme vous le savez, un "sprite" est donc un objet graphique que l'on peut déplacer finement sur une image, par dessus un background, sans altérer celui-ci, tel un spectre, un esprit. Le terme sprite "hardware" implique qu'il y a une fonction hardware qui va prendre cela en charge à l'inverse du sprite "software" (principalement utilisé sur les micro de l'époque, ZX Spectrum, CPC, Atari ST…) que l'on déplace, pixel par pixel, par l'intermédiaire du CPU, dans un framebuffer, et qui nécessite tout un processus lourd pour restaurer ensuite le background qui a été altéré à son passage. On crée alors l'illusion d'un sprite par un traitement relativement lourd, ce que permet d'éviter le sprite hardware.

Le terme "sprite" a été semble t'il inventé par l'un des ingénieurs Texas Instrument du TMS9918. Dans un interview de 1992, lors d'une conférence Texas Instrument, Karl Guttag, ingénieur sur le TMS9918, l'évoque brièvement: *".. lot of people when they think of TI and graphics, think of the 9918 and sprites (the term "sprites", a greek fairy, was coined by one of the 9918's definers)"* . Avant cela on utilisait plutôt le terme MOB pour "mobile object."
Le TMS9918 n'est pas n'importe quel chip. C'est le chip graphique que Texas Instrument a conçu en 1978-1979 pour ses micros TI99/4 et qui intégrait une fonction de sprite hardware relativement avancé. En intégrant ce chip à son catalogue tel un chip générique, au côté de ses célèbres sound chip PSG, Texas Instrument a ainsi popularisé et standardisé l'usage de sprite hardware dans le salon. En effet ce TMS9918 s'est retrouvé dans beaucoup de machines dont la Colecovision, le MSX ou la première console de Sega (SG-1000) puis a servi d'inspiration pour la NES, la SMS.

Tout ça crée un lien assez direct entre le terme "sprite" et le concept même de sprite hardware même s'il y a probablement eu du sprite hardware avant le TMS9918 en arcade (mais plus difficile à explorer). Sans compter que l'Atari VCS en proposait aussi en 1977 mais sous une forme plus discutable. Si la VCS avait bien un chip graphique (TIA) capable de prendre en charge de façon hardware un background et 2 sprites (voir 5), il ne le faisait que sur une seule scanline. Le TIA était en quelque sorte un GPU 1D si bien que le véritable chipset graphique de la VCS c'est la combinaison du CPU + TIA qui permet alors de construire une image 2D. Donc un cas un peu en marge de la préhistoire du sprite hardware. Mais on pourrait citer aussi l'Odyssey 2 (Videopac en Europe) qui en 1978 embarque un chip graphique Intel qui supporte 4 sprites hardware 8x8

(mais sans réel support de background) ou le chip Signetics PVI 2636 et ses 4 sprites hardwares 8x10, qui a équipé certaines consoles marginales tel que la 1292 APVS, l'Interton VC400 ou l'Arcadia 2001 et autres clones. Un chip qui a des similitudes troublantes avec celui de l'Odyssey 2 (je ne sais pas quelle filiation pourrait exister entre les 2). Autant de machines que j'ai préféré ne pas traiter ici pour débuter directement avec l'Intellivision qui est une machine déjà assez moderne et proche de tout ce qui va suivre ensuite.

Il me reste à évoquer un dernier point important pour comprendre ce billet. L'usage courant du terme "sprite" est un peu différent de celui que je peux faire dans ce type de billet. En effet, dans l'usage courant, on associe le terme "sprite" à un personnage, un monstre, un véhicule... mais un personnage est quasi systématiquement composé d'un assemblage de plusieurs sprites hardware. Un personnage sur NES, aussi bien que sur CPS, peut être composé d'une vingtaine de sprites hardware voir plus. Pour cette raison, chaque fois que je parle de sprite dans ce billet, je fais référence aux sprites hardwares, et pour évoquer l'objet graphique, le personnage à l'écran, j'utilise plutôt le terme "Metasprite".

Le Tableau comparatif

Vous pouvez cliquer directement dans le tableau sur la machine ou le critère pour rejoindre le paragraphe correspondant et revenir au tableau en cliquant sur le titre du paragraphe.

	Year	Pixel-sprite per scanline	Bit-sprite per scanline	Scanline coverage	Max pixel-sprite per frame	Max screen coverage	Sprite per frame	Colors per sprite	Sprite layer colors	Resolution class
Intellivision	1980	64	64	40%	1024	3,3% or 6,6%	8	1	8	1 or 2
Commodore 64	1982	96 or 192	192	60%	2016 or 4032	6,3%	8	1 or 3	10	2.1 or 4.2
Coleco / SG1000 / MSX	1982/3	64	64	25%	8192	16,7%	32	1	15	3.2
Super Cassette Vision	1984	368 to 776	368 to 776	192% to 404%	81 920	192%	128	1	16	2.8
NES	1983	64	128	25%	8192	13,3%	64	3	12	4
Master System	1985	64	256	25%	8192	16,7%	64	15	15	3.2
Game Boy	1989	80	160	50%	5120	22,2%	40	3	4	1.5
MSX 2	1985	128	128	50%	8192	15,1%	32	1 to 16	16	3.2
GX-4000	1990	256	1024	80% or 160%	4096	5,3% or 10,7%	16	15	15	2.5 or 5
PC-Engine	1987	256	1024	73% or 100%	61 440	73% or 100%	64	15	240	4 or 5.5
Megadrive	1988	256 or 320	1024 or 1280	100%	61440 or 76800	100%	64 or 80	15	60	4 or 5
SuperGrafX	1989	512	2048	145% or 200%	122 880	145% or 200%	128	15	240	4 or 5.5
SNES	1990	272	1088	106%	65 280	106%	128	15	120	4
Neo-Geo	1990	1536	6144	480%	344 064	480%	380	15	3840	4.7
Sega System16	1986	800	3200	250%	179 200	250%	128	14	960	4.7

Capcom CPS1	1988	4096	16 384	1067%	65 536	76,2%	256	15	480	5.6
Sharp X68000	1987	512	2048	133%	32 768	38,1%	128	15	240	5.6

Glossaire

Quelques explications sur les 9 critères techniques que j'ai sélectionné pour le tableau comparatif.

Pixel-sprite per scanline
Il s'agit du nombre de pixel de sprite que la machine peut afficher sur la même scanline (la même ligne horizontale de l'écran). Si la machine peut afficher 8 sprites par scanline qui font chacun 8 pixels max de large (NES, SMS) ça donne alors 8x8 = 64 pixel-sprite par scanline. C'est un gros indicateur sur les performances internes de la machine car la difficulté de traitement des sprites est souvent là. Réussir à traiter beaucoup de pixel-sprite dans le délai très court du balayage d'une scanline (environ 0,06 milliseconde). A chaque scanline la machine va parcourir toute la liste de sprite pour déterminer ceux visible à la scanline suivante et charger les pixel-sprite concerné. Quand le jeu dépasse cette limite alors il y a "overflow" et les pixel-sprite suivant ne seront pas affichés. C'est cet affichage partiel des sprites qui potentiellement va générer du flickering (ou une disparition selon le choix du programmeur).

Bit-sprite per scanline
C'est à peu près la même chose que les pixel-sprite par scanline mais en prenant en considération la profondeur des pixels (les couleurs). Une profondeur de 1 bpp (1 bit par pixel) correspond à des pixel-sprite monochromes. 2 bpp correspond à des pixel-sprite qui disposent d'un choix parmi 3 couleurs. 4 bpp pour 15 couleurs. Ce critère complémentaire permet par exemple de distinguer des machines comme le MSX, la NES et la Master System qui ont les mêmes valeurs de pixel-sprite par scanline mais pas la même profondeur (respectivement 1, 2 et 4 bpp qui serviront donc de multiplicateur) et donc pas les même liberté en terme de couleur. On ne peut pas considérer à équivalence un MSX avec ses sprites monochromes (qui souvent vont en plus être superposés pour simuler des sprites multicolores et donc atteindre d'autant plus vite l'overflow) et une Master System avec ses sprites nativement 15 couleurs. C'est donc une mesure de performance probablement plus globale encore que les pixel-sprite par scanline.

Scanline coverage
Une valeur qui exprime quelle proportion de la scanline, ou largeur d'écran, la machine peut couvrir avec ses sprites. C'est le rapport entre la valeur de "pixel-sprite per scanline" et la résolution horizontale de l'image. Ça permet d'estimer si on peut aligner horizontalement beaucoup d'objets et/ou avoir des objets larges, avant de subir des disparitions ou du flickering. Par exemple, les MSX, NES ou SMS ont des valeurs faibles dans ce domaine (25%) qui expliquent que ces machines sont tristement connues pour leur flickering de sprite. Lorsque l'on essaye de remplir plus d'un quart de la largeur de l'écran avec des sprites alors on dépasse déjà la limite de ces machines. Bien sûr, plus la résolution de la machine est faible, plus il sera facile d'avoir un bon taux de couverture. C'est ce qui permet par exemple à l'Intellivision ou à la Gameboy d'avoir de meilleurs chiffres dans ce domaine grâce à leur résolution horizontale faible de 160 pixels. Et pour cette raison c'est une information complémentaire intéressante.

Max pixel-sprite per frame
Un indicateur pour spécifier combien de pixel-sprite la machine peut potentiellement afficher sur la même frame si on réunit toutes les bonnes conditions. C'est a dire si on priorise les plus grosses dimensions de sprite (sauf magnificence) et que l'on répartit de façon optimale les sprites sur l'écran pour contourner l'overflow par scanline. Une valeur qui va donc être impactée par les formats de sprite proposés (taille de sprite) et la taille de la liste de sprite (combien de sprite) ce qui donne une mesure de performance plutôt globale. Les machines sans multiplexage hardware des sprites (Intellivision, C64, GX4000) sont naturellement handicapé sur ce critère puisqu'elles proposent peu de sprites à l'écran.

Max screen coverage
Comme pour le "scanline coverage", il s'agit de mettre en relation la valeur précédente (max pixel-sprite par frame) avec la résolution globale de l'image pour estimer quelle est la capacité de la machine à couvrir son écran de sprite. Ca mesure donc des performances globales d'affichage de sprite mais cette fois à l'avantage des machines à faible résolution qui peuvent ainsi plus facilement couvrir leur écran de sprite et/ou afficher de gros sprites, grâce à leurs pixel-sprite nativement plus gros là ou les machines à résolution élevée vont être plus handicapé. C'est donc une mesure complémentaire et plus proche du ressenti, de ce qu'on voit concrètement à l'écran. Mais pour les mêmes raisons que précédemment, les machines sans multiplexage hardware sont particulièrement handicapées ici.

Sprite per frame
Ça parle de lui-même. C'est le nombre total de sprites hardware affichables par frame. Sans aucun doute le critère le plus connu et partagé des joueurs qui lui donnent probablement plus d'importance que mérité. Quand il est isolé, il ne donne pas beaucoup d'informations. Ce n'est pas tant une mesure de performance d'affichage brute qu'une mesure du multiplexage hardware et de la capacité à trier rapidement une grosse liste de sprites.
La capacité brute du hardware dans l'affichage de sprite transparaît plutôt dans la valeur de pixel-sprite par scanline. C'est ensuite l'ajout d'un multiplexage hardware qui va permettre de faire grossir artificiellement la liste de sprite. Cela consiste en quelque sorte à exploiter le temps d'affichage de l'image et donc de balayage vertical (qui est lent à l'échelle électronique) pour recycler, d'une scanline à l'autre, les slots de sprite hardware (ca peut etre de la mémoire, de la bande passante…) nécessaire à chaque sprite qui doit être affiché. Dès qu'un slot se libère au fil du balayage, on peut le mettre à disposition d'un prochain sprite plus bas dans l'image. C'est le TMS9918 de Texas Instrument (Coleco / SG1000 / MSX) qui semble avoir initié cette méthode de multiplexage hardware des sprites qui permet donc de décorréler le nombre de sprite par scanline (qui est la vrai limite physique du hardware) du nombre de sprite par frame. Cette méthode est ensuite devenue un standard.
Les machines du tableau qui n'ont pas de multiplexage hardware comme l'Intellivision, le C64 ou la GX4000 sont forcément à la traîne sur ce critère (leur nombre de sprites par frame est égale à leur nombre de sprites par scanline). En réalité, la différence entre une Intellivision et une NES (ou SMS) par exemple, c'est principalement cette absence de multiplexage. Les 2 ont la même capacité brute de seulement 8 slots internes de sprites hardware mais cette dernières va recycler cette capacité pendant toute la durée du balayage vertical pour gonfler la liste jusqu'à 64 sprites (tout comme une PC-Engine qui passe de 16 à 64 sprite, une MD de 20 à 80 sprites ou une SNES de 32 à 128 grâce à ce multiplexage).

Colors per sprite
C'est simplement le nombre maximum de couleurs différentes que l'on peut utiliser dans un même sprite hardware. C'est directement lié à la profondeur des pixel-sprite que j'ai évoqué dans le paragraphe sur les bit-sprite par scanline. Attention, les machines qui doivent se contenter de sprite monochrome font souvent usage de superposition de plusieurs sprites hardware de différente couleur pour simuler un metasprite multicolore donc ne vous étonnez pas de voir des sprites multicolores sur ces machines monochrome. Bien sûr ce type de superposition va impacter négativement les performances globales d'affichage de sprite. On

retrouve aussi cette pratique pour des machines qui utilisent des sprites 3 couleurs comme la NES, souvent pour le sprite du joueur (notamment Capcom).

Sprite layer colors
Cette fois c'est le nombre maximum de couleurs potentiellement affichable par le layer sprite entier, c'est à dire en cumulant tous les sprites. Pour certaines machines ce nombre correspond à la totalité des couleurs de la machine et pour d'autres ce n'est qu'une portion de celles-ci. Donc il ne faut pas interpréter excessivement ce nombre. Tout comme il ne donne pas d'indication non plus sur la gamme de nuance à disposition (la précision de codage des couleurs). Pour ces détails il faut lire le paragraphe associé à chaque machine.

Resolution class
Il s'agit de donner une indication sur la finesse des sprites. Si la résolution est un handicap pour le scanline coverage et screen coverage, ça n'en est pas moins un critère de qualité tout de même qu'il faut bien spécifier et ainsi mieux comprendre certaines autres valeurs du tableau. Des sprites fins et détaillés, c'est chouette. Je l'indique donc ici sous une forme simplifiée en prenant comme unité la résolution de l'Intellivision (160x96). Une valeur de 4 veut dire 4x plus de pixel qu'un background Intellivision et donc une densité de pixel-sprite 4x supérieurs si jamais on superposait exactement toutes les images des machines en les déformant un peu pour compenser les différences de format. J'ai hésité entre cette quantification et une autre approche qui consisterait à parler plutôt de dotcklock, qui est une autre façon de parler de densité de pixel, mais ça me parait plus pertinent comme ça.

Annotations

Je vais ici commenter en partie les chiffres du tableau, machine par machine, et donner quelques explications pour mieux comprendre certains d'entre eux qui peuvent surprendre ou simplement pour compléter avec des informations qui ne peuvent pas transparaître dans le tableau. C'est plutôt pour un public avertie qui aime les détails techniques et dans la limite de mes connaissances sur ces machines.

Intellivision
Le chip graphique de l'Intellivision, développé par General Instrument, était plutôt avant-gardiste pour 1979 (il offre même plus de possibilité de scrolling qu'un MSX) notamment pour son intégration de sprites hardwares, à l'image de son concurrent direct, Texas Instrument et son TMS9918 conçu à la même époque pour leurs micros (mais qui deviendra bien plus célèbre par la suite).
On note des chiffres de "screen coverage" handicapé par l'absence de multiplexage hardware des sprites qui condamne la machine à une liste de sprites faible (8 sprites à l'écran). Par contre le "scanline coverage" est plutôt bon, meilleur que sur nos NES ou SMS (mais monochrome), en partie grâce à sa faible résolution (160x96), la plus faible de toute la liste. Mais cette résolution est avant tout celle du background. On peut aligner la résolution des sprites sur cette résolution grossière mais il est possible aussi de doubler leur finesse verticale ce qui explique les 2 chiffres de "screen coverage" selon la finesse des sprites (en effet les sprites mieux résolus peine encore plus à remplir l'écran). Le format max de sprite étant 8x16 (sinon 8x8) comme sur NES et SMS. Il y a une feature assez poussée de gestion hardware des collisions de sprites. C'était plutôt courant dans les premières consoles mais cette fois si c'est sensiblement plus avancé et utile que d'habitude. Il y a 81 flags dédiés juste à cette fonction. Ça permet par exemple dans la version Intellivision du très célèbre River Raid de proposer au joueur de slalomer entre les centaines d'arbres du décors avec des collisions au

pixel, sans hitbox. Une proposition unique à cette version du jeu. En ce qui concerne les options de magnificence de sprite, je vous renvoie à l'introduction du document.

Commodore 64

Le C64 est un micro qui a eu le droit à un vrai chip graphique custom, le VIC-II, qui fait du C64 l'un des micro les plus typés console. Un chip développé par MOS Technologie (racheté par Commodore) qui sont les concepteurs du célèbre CPU 6502 qui équipe entre autres le C64 lui-même donc une bonne synérgie entre les composants. Comme pour l'intellivision, le C64 a un "screen coverage" très impacté par l'absence de multiplexage hardware des sprites qui bride la machine à seulement 8 sprites. Mais à l'inverse de l'Intellivision, le hardware de la C64 permet aux programmeurs de mettre en place manuellement un multiplexage software des sprites, et ce n'est pas si fréquent comme possibilité. Ça permet donc un recyclage des sprites hardwares au fil du balayage vertical de l'écran (une forme de "raster effect" sur les sprites) à condition d'une certaine virtuosité dans la programmation. Ainsi, un même sprite hardware peut être utilisé plusieurs fois sur la hauteur de l'image. On peut même dire que les limites du hardware (8 sprites c'est peu pour faire un jeu d'action) poussent les programmeurs dans cette direction ce qui donne une majorité de jeu C64 qui use de ce stratagème. C'est de très loin je pense la machine qui pratique le plus le multiplexage software de sprite. Il existe quelques consoles qui le permettent comme la Megadrive ou la Gameboy mais la présence de multiplexage hardware rend cette alternative peu nécessaire et quasi inutilisée dans les jeux (cf. Sonic 2 pour le split-screen par exemple). Le multiplexage software, par définition, ne fait donc pas partie du support hardware natif et donc du tableau, en plus de ne pas être vraiment quantifiable. Pour cette raison ce chiffre de "screen coverage" n'est pas très représentatif des jeux C64.
Les valeurs doubles présentes dans les cases de pixel-sprite par scanline et par frame (ainsi que pour les couleurs) s'expliquent par la présence d'un mode "hires". On peut donc choisir (indépendamment pour les sprites et le background) entre une résolution 160x200 et 320x200 pixels, même si bien sûr le 160x200 va être majoritairement utilisé pour les jeux. Un sprite hardware C64 est donc soit hires 24x21 monochrome ou lowres 12x21 mais 3 couleurs. Les 2 ayant la même taille apparente, Il s'agit juste d'une différence de densité (et donc ça n'a pas d'impact sur le scanline et screen coverage). La quantité de data reste la même pour les 2 types de sprite. Ca revient simplement à choisir entre couleur ou résolution, sachant qu'on peut mixer les 2 types de sprite sur une même scène et un même metasprite (souvent un personnage lowres en couleur sur lequel on ajoute des contours noirs en hires). Malgré les 3 couleurs possibles par sprite, les contraintes sont plus fortes que sur NES car il y a 2 couleurs qui sont communes aux 8 sprites. On ne peut choisir que la 3ème couleur individuellement à chaque sprite (et parmi une gamme de nuance globale très limitée de 16 couleurs). Mais comme pour le multiplexage, rien n'empêche de faire des modifications au raster avec le CPU pour offrir artificiellement un peu plus de liberté mais ça n'entre pas dans le cadre du support hardware natif et donc de ce tableau. Les 8 sprites sont de vrai slot hardware interne et donc peuvent être cumulés sur la même ligne ce qui, contrairement au screen coverage, offre un très bon scanline coverage pour l'époque, même en "hires".
En ce qui concerne les options de magnificence de sprite, je vous renvoie à l'introduction du document.

ColecoVision / SG-1000 / MSX

Ces 3 machines (et d'autres) partagent le même chip vidéo générique de Texas Instrument, le TMS9918A, et donc les mêmes caractéristiques graphiques (et pas seulement puisqu'elles ont aussi le même CPU). Comme évoqué en introduction, le TMS9918 (1979) est une petite révolution car en intégrant des sprites hardware dans un chip générique du catalogue TI, il permet de généraliser le principe de sprite hardware à un large panel de machine ordinaire sans avoir besoin de faire de grosse R&D pour développer des chips custom. Et en plus il le fait plutôt bien puisque à l'inverse de l'Intellivision, C64 ou GX4000, le TMS9918 propose nativement un multiplexage hardware des sprites qui permet d'obtenir une liste très confortable de 32 sprites à partir de seulement 4 slots internes. 4 sprites par scanline (et monochrome) ce n'est pas beaucoup mais les sprites du TMS9918A supporte le format 16x16 pixels (contre 8x16 max sur NES ou SMS), ça compense en partie. En ce qui concerne les options de magnificence de sprite, je vous renvoie à l'introduction du document.
Je n'ai pas grand chose à ajouter sur tous ces chiffres si ce n'est qu'ils sont globalement bien équilibrés. La résolution est de 256x192 comme une SMS. C'était un peu la HD de l'époque car les consoles avait plutôt une résolution horizontale de 160 pixels (pour avoir un dotclock 3.58mhz équivalent au color-clock du signal

NTSC). Le TMS9918 va initier le dotclock 5.37mhz qui va devenir un standard en perdurant jusqu'à la SNES et implique des pixels un peu anamorphiques, plus large que haut. Le plus gros manque du côté des sprites c'est l'absence de flipping (symétrie hardware, très utile pour les sprites car un même sprite, par exemple un personnage, a souvent plusieurs orientations dans un jeu) et l'absence de priorité basse pour pouvoir placer les sprites derrière le background (il manquait surtout du scrolling hardware au TMS9918 pour en faire un chip vidéo complet pour le jeu vidéo mais en 1979 on ne peut pas tout avoir).
Le TMS9918 c'est avant tout une étape importante dans l'histoire du sprite hardware d'autant plus quand on sait que le terme "sprite" est né de la bouche des ingénieurs TI de ce TMS9918 semble t'il. Il va standardiser un certain nombre de choses.

Super Cassette Vision
Une machine du Japonais Epoch qui succède à la Cassette Vision (au même titre que la Super Famicom succède à la Famicom ^^) en 1984. Une console très particulière de cette liste puisque celle-ci a tout misé sur les sprites hardwares justement. Un cas singulier qui nécessite de s'y attarder un peu plus que d'autres. Autant son hardware a beaucoup de lacune et de retard dans divers domaines pour une machine sortie un an après la Famicom (résolution inférieur à une Coleco avec une palette aussi pauvre, sprite monochrome, un seul canal audio, la quantité de RAM d'une VCS…) mais des performances en sprite hardware surprenante avec un pipeline peu conventionnel. La SCV choisit une stratégie proche de la NeoGeo. Elle n'a pas vraiment de layer background donc les sprites vont servir aussi à construire celui-ci. Pour cette raison, il y a beaucoup de ressources sprite sur SCV, aussi bien pour alimenter le background que pour compenser la monochromie des sprites en permettant d'en superposer plusieurs couches (le mode "bicolor" n'étant qu'une automatisation de la superposition de sprite).
L'une des raisons de ses performances singulières vient du fait que la SCV semble à priori exploiter toute la durée de la scanline (avec un double line-buffer) pour traiter les sprites, là ou les autres consoles se contentent en général de faire ce lourd traitement pendant la courte durée du Hblank (le petit délai off screen entre le balayage de 2 scanlines) afin de laisser le reste du temps disponible au traitement du/des layer background et partager ainsi le bus VRAM entre les différentes tâches graphiques. A l'inverse des autres consoles, la VRAM de la SCV, et donc son bus, est entièrement dédiée au chargement des pixel-sprite (comme la NeoGeo avec sa C-ROM) car tout le reste se trouve dans des mémoires interne au chip graphique. Mais d'après mes tests, la plus grande particularité du pipeline des sprites SCV semble être l'absence d'une phase de tri des sprites à chaque scanline qui soit dissociée et exécutée en parallèle du chargement des pixels-sprite comme c'est le cas dans les autres consoles. La SCV s'abstient en quelque sorte de faire ce tri en amont pour simplifier le processus étant donné la grosse liste à traiter (128 sprites). Mais cette méthode implique que plus le sprite à traiter est haut dans la liste, plus il est coûteux à traiter, ce qui donne un nombre de pixel-sprite par scanline (et donc une limite d'overflow) qui varie selon le contexte, selon les sprites concernés par la prochaine scanline à traiter. C'est pour cette raison que certains chiffres du tableau concernant la SCV sont plutôt des intervalles entre une valeur min et max. C'est d'ailleurs la seule machine du tableau dans ce cas.
Ces chiffres, je les ai obtenus à partir de tests d'affichage sur la console mais qui ont été fait sur une SCV Yeno française PAL. Je n'ai pas pu encore trouver quelqu'un pour les faire aussi sur une SCV japonaise (si vous connaissez quelqu'un ça m'intéresse) mais il n'y a pas de raison que les chiffres soient très différents. Ces chiffres sont très impressionnants, sensiblement supérieurs aux consoles 16 bit. Évidement faut relativiser en rappelant que ce sont des pixel-sprite monochromes et que la résolution ici est de 192x222. Mais si on compare par exemple à un MSX qui utilise aussi des sprites monochrome, on est exactement sur un x10 en pixel-sprite par frame et plus ou moins sur les autres valeurs aussi mais le MSX a un vrai layer background (avec certaine liberté de couleur) pour compenser en partie. Et la construction de la liste de sprite de la SCV doit être un sacré défi car construire une liste de 128 sprites avec le peu de ressource de la machine, ça ne doit pas être simple. D'autant que la liste de sprite (512 octets) fait 4 fois la taille de la RAM (128 octets) ce qui implique qu'on ne peut pas la double-bufferiser et construire tranquillement la liste en RAM avant transfert comme sur les autres machines. Il faut directement taper dans le cache interne du chip graphique alors même qu'il est utilisé pour l'affichage en cours. Sacré casse-tête. Donc une machine à la fois très surprenante et bancale.

NES

La NES a bénéficié d'un vrai chip graphique custom sur mesure (par Ricoh) mais qui s'est très probablement inspiré du TMS9918 étant donné certaines similitudes. On sait que Nintendo était assez proche de Coleco et se reconnaissait dans cette entreprise au parcours très similaire jusque là. Ils étaient même partenaires (pistolet optique Telstar, Donkey Kong Colecovision…), Coleco voulant même que Nintendo distribue la Colecovision au Japon. La vision des prototypes de Colecovision fait sans doute partie des déclencheurs pour accélérer le projet Famicom. Nintendo a très probablement décortiqué la Colecovision et son TMS9918 pour s'en servir d'objectif à dépasser.

Les sprites hardwares de la NES ont les mêmes caractéristiques que ceux de la SMS. On est sur 2 machines quasi jumelle sur ce point, excepté que les pixel-sprite de la NES ont une profondeur 2 fois moindre (2 bpp vs 4 bpp) ce qui donne des contraintes fortes de couleurs (seulement 3 couleurs par sprite). Ces contraintes peuvent parfois pousser les devs a faire des superpositions de sprite comme sur les machines précédentes, pour augmenter artificiellement le nombre de couleur des sprites (Capcom le fait souvent pour le sprite du joueur: Ducktales, Megaman…). Cette méthode va donc surcharger encore plus la scanline et atteindre encore plus vite la limite qui produit du flickering, sachant que ce sont déjà des machines qui ont des valeurs de scanline coverage faibles. Heureusement ce type de superposition n'est pas non plus la norme sur NES, la plupart des sprites se contentant des limites de 3 couleurs.

En proposant une liste de 64 sprites, la NES introduit un nouveau standard de quantité de sprite plutôt confortable pour construire une scène de jeu. C'est donc maintenant la faible quantité de pixel-sprite par scanline qui devient vraiment le centre des préoccupations et le principal défi. Les formats de sprites sont limités et exclusifs (le format choisi s'applique à tous les sprites à l'écran). Il y en a que 2: 8x8 ou 8x16 comme sur SMS. Par contre la NES est en quelque sorte encore plus focus sur les sprites que la SMS avec notamment du flipping hardware, une gestion de priorité avec le background qui permet certain tricks (et qui se gère donc dans les attributs des sprites plutôt que dans ceux des tuiles background sur SMS), la possibilité de pouvoir piocher dans l'intégralité du set de tuile, et un DMA dédié au chargement de la liste de sprite. En effet la NES intègre une fonction câblé pour accélérer le chargement de la liste de sprite (256 octets) dans le cache interne du chip graphique. Une tâche qui doit être exécutée à chaque frame et qui ici ne prend que 512 cycles CPU soit 1,7% des ressources CPU (1,5% en PAL) grâce à cela.

L'une des forces de la NES est aussi d'avoir son bus graphique câblé directement sur le port cartouche ce qui permet au chip graphique d'accéder directement aux tuiles qui sont dans la cartouche, comme une NeoGeo, et combiné à du bank switching ca permet de simplifier l'animation des sprites et d'offrir certaine possibilité supplémentaire.

Les sprites NES se partagent 4 palettes (de 3 couleurs chacune) qui leur sont dédiées soit au mieux 12 couleurs pour l'ensemble des sprites à l'écran mais souvent il y a 1 ou 2 couleurs redondantes entre les palettes ce qui donne plutôt une dizaine max en pratique. Les couleurs sur NES ne sont pas au format RGB, c'est un format chrominance/luminance codé sur 6 bit qui offre 54 nuances. On peut remarquer que le chiffre de screen coverage est un peu plus faible que sur SMS ou Coleco / SG1000 / MSX. C'est parce que la NES utilise nativement une résolution 256x240 plus étendue verticalement et donc plus difficile à couvrir que le 256x192 de ces machines mais en réalité la totalité des sprites NES peuvent donc couvrir exactement la même surface sur l'écran que ses concurrentes.

Master System

Un chip graphique qui se veut une évolution custom du TMS9918 (une branche évolutive distincte de celle choisie par Yamaha pour les MSX2 et MSX2+) et même rétrocompatible avec celui-ci pour pouvoir supporter les jeux SG-1000. J'ai déjà évoqué certains éléments sur le paragraphe NES donc je vais essayer de ne pas trop me répéter. On est sur des caractéristiques très similaire a la NES mais la force de la SMS, et ce qui la distingue, est surtout d'avoir introduit pour la première fois sur console une profondeur de pixel de 4 bit (15 couleurs), aussi bien pour le background que pour les sprites, et qui permet enfin de relâcher un peu les fortes contrainte locals de couleur qu'il y avait sur les consoles jusque là. Le 4 bpp (bit par pixel) devient un standard sur console (voir en arcade) qui va tenir jusqu'à la NeoGeo, elle-même 4bpp. C'est l'apport principal de la SMS.

Ça ne veut pas dire qu'il n'y a pas de forte contrainte quand même sur les couleurs des sprites car tous les sprites doivent se partager la même unique palette 15 couleurs. Une palette à construire en piochant dans un choix de nuances qui reste limité aussi par son RGB 6 bit et 64 nuances. Mais ce sont des contraintes qui deviennent plus globales et diluées là où elles étaient fortes et locales sur NES. Ça permet aussi à la SMS de s'affranchir de cette stratégie coûteuse de superposition de sprite qu'on retrouvait sur les machines précédentes (NES inclus) pour augmenter artificiellement le nombre de couleurs locales. Et c'est une bonne chose pour éviter de surcharger la scanline car ça reste une machine avec un scanline coverage faible (flickering de sprite). La même limite de 8 sprites par scanline que la NES. Ce sera la dernière console aussi faible en scanline coverage. D'ailleurs la SMS va parfois profiter de sa profondeur de pixel 4 bpp pour s'autoriser l'utilisation de sprites softwares (construit au CPU tel un CPC) afin de contourner la limite de sprite par scanline tel qu'on peut le voir dans Golden Axe par exemple (qui, de ce fait, n'a pas de flickering), mais ca donne un jeu entre 7 et 10 fps (les sprites hardwares servent justement à éviter ca).
Les sprites SMS sont un peu handicapé aussi par leur filiation avec le TMS9918. L'absence d'attributs de sprite (autre que les indispensables X,Y et ID tile) implique quelques lacunes comme l'absence de flipping (symétrie verticale et horizontale) qui est le plus embêtant pour les sprites qui ont souvent besoin d'être orientables dans plusieurs direction. Les vrais commodités sur SMS sont plutôt du côté du layer background ce qui notamment rend souvent plus facile que sur NES l'interaction des sprites avec le background quand on veut les faire passer derrière celui-ci. Par contre on peut accéder à la VRAM (et donc à la liste de sprite ou au set de tuile des sprites) même pendant le balayage de l'écran. Ça peut être une commodité utile.
La résolution reste celle du TMS9918, c'est à dire 256x192 (la SMS 2 propose un mode alternatif 256x224 mais utilisé seulement sur quelques jeux PAL Codemaster). C'est un format qui commence à être étriqué verticalement pour l'époque notamment pour du shmup vertical ou si le HUD est imposant.

Game Boy

C'est la seule console portable du panel. La plus emblématique. Les sprites de la GB ont des caractéristiques assez similaires à la NES. Un choix de format 8x8 ou 8x16, des pixels 2 bpp, du flipping, mais jusqu'à 10 sprite sur la même scanline (vs 8), ce qui combiné à la faible résolution de la GB (160x144) permet à cette dernière d'avoir ce très bon chiffre de scanline coverage que l'on voit dans le tableau. Le double des consoles 8 bit canoniques. La faible résolution de l'écran est aussi responsable de ce sentiment que les sprites sur GB sont globalement plus gros que sur NES. Mais en réalité, pour les portages notamment, ce sont souvent les mêmes sprites. L'illusion vient de cette faible résolution qui permet au moindre petit sprite d'occuper une surface importante de l'écran.
Au-delà du gros manque évident de nuances (4 nuances de gris) qui est problématique, le plus gros défaut des sprites GB semble être la gestion des priorités inter-sprites. La priorité entre les sprites se fait simplement selon leur coordonnée X sur l'écran et pas selon l'ordre dans la liste ce qui réduit fortement le contrôle des priorités d'affichages des sprites (pouvoir choisir quel sprite apparaît devant quel autre). De ce fait, on comprend mieux la nécessité d'un bon scanline coverage car la gestion de l'overflow par l'intermédiaire du flickering nécessite un bon contrôle des priorités qui semble plutôt compromis sur GB. Par contre la GB fait partie de ces machines qui autorisent le multiplexage software des sprites qui peut s'ajouter au multiplexage hardware pour gonfler un peu plus le nombre de sprites artificiellement mais sans doute anecdotique comme usage comme pour la Megadrive.

MSX 2

Le MSX 2 utilise une évolution du TMS9918 par Yamaha, une sorte de branche évolutive parallèle à celle de Sega pour sa SMS. La partie sprite du MSX 2 est relativement similaire à celle du MSX 1. Toujours 32 sprites 1 bpp et des formats 8x8 ou 16x16 mais les performances de pixel-sprite par scanline (et donc de scanline coverage) sont doublées grâce à la possibilité d'aligner 8 sprites par scanline au lieu de 4. La question des couleurs est plus complexe d'où l'intervalle un peu vague dans le tableau. Malgré que les sprites gardent un format 1 bpp plutôt décevant pour cette époque (la SMS est passé au 4 bpp), le MSX 2 offre différentes solutions pour éviter justement d'avoir des sprites monochromes qui font tache. 2 solutions pour être exact. La première consiste à proposer une "color table" pour chacun des 32 sprites qui permet de changer l'unique couleur du sprite à chacune de ses lignes (donc jusqu'à 16 couleurs pour le format 16x16). Il s'agit simplement

d’adapter aux sprites une méthode qui était déjà utilisée sur MSX 1 mais pour le background (mode graphics 2). Sauf que ces “color table” sont associées directement aux slots des sprites, pas aux set de tuile comme pour les background du mode graphics 2, ce qui rend l’usage plus compliqué notamment si le sprite est animé ou si on veut changer les priorités. Et puis on ne contrôle pas la couleur individuel des pixels, juste par ligne. Pour ces raisons, la valeur du tableau qui indique jusqu'à 16 couleurs par sprite est une valeur théorique max qui ne reflète pas la réalité effective de son usage.
Le plus intéressant est probablement l’autre feature de “color mixing”. Comme sur MSX 1, les sprites monochrome du MSX 2 incite à pratiquer la superposition de sprite pour augmenter artificiellement le nombre de couleurs, sauf que cette fois une fonction hardware permet d’exploiter pleinement cette méthode. Superposer 2 sprites ne permet plus d’avoir 2 couleurs mais 3, comme si on manipulait un vrai sprite 2 bpp (NES) car le MSX 2 va appliquer un OR logique entre les pixels des 2 sprites (et donc avec un vrai contrôle individuel de la couleur des pixels, pas juste par ligne). Ainsi la superposition de 2 sprites devient un peu la norme sur MSX 2 (d'où la nécessité d’avoir doubler le nombre de pixel-sprite par scanline pour préserver un scanline coverage effectif au moins similaire à une NES/SMS) et si combiné avec les color tables on peut obtenir de jolies sprites entre la NES et la SMS mais ca demande une gymnastique pas simple.
Toujours 16 couleurs à disposition des sprites, comme sur MSX, mais avec une précision de codage des couleurs qui passe de 3 bit à 9 bit, donc plus que sur SMS (6 bit, 64 nuances), équivalent à ce qu’on trouve sur PCE, soit une gamme de 512 nuances dans lesquelles piocher ces 16 couleurs. J’ai pris le 256x212 comme résolution de référence. Donc un peu plus de ligne que sur le TMS9918 du MSX1 d'où un chiffre de screen coverage un peu plus bas. Je n’ai pas pris en considération les modes hires 512x212 pas très pertinents et adaptés pour les jeux.

GX-4000
La GX-4000 de Amstrad est plus ou moins un CPC-Plus consolisé qui lui même est un CPC boosté avec quelques fonctions hardware supplémentaires dont la principale est justement l’ajout de sprite hardware. 16 sprites hardware 4 bpp dans un format unique de 16x16 pixels. Mais c’est une implémentation un peu archaïque (sans doute parce que c'est un ajout par dessus un vieil hardware déjà existant avec lequel il a fallu composer). Déjà il n’y a pas de multiplexage hardware des sprites d'où ce faible nombre de sprite par frame qui est donc aussi le nombre de sprite par scanline ce qui, en contrepartie, permet ces très bon chiffre de pixel-sprite par scanline et de scanline coverage. Pas de flipping non plus ni choix de format.
Mais la chose la plus singulière (et handicapante) par rapport aux autres machines c’est l’absence d’un set de tuile à disposition des sprites. Sur les autres machines les patterns des sprites sont stockés sous la forme d’un catalogue dans une RAM externe et offrent donc un large choix. A l’inverse sur GX-4000 les patterns des sprites sont toutes stockés dans le petit cache interne au chip graphique. Il y a juste ces 16 slots de sprites internes dans lesquels sont chargés 16 patterns de 16x16 pixels a la manière d’une Odyssey 2 / Videopac de 1978. Ça implique que si on veut animer le sprite ou flipper le sprite ou changer la priorité, il faut modifier et charger manuellement l’intégralité de la pattern. Un chargement qui en plus est particulièrement mal optimisé car il se fait au CPU pixel par pixel (donc une écriture 8 bit permet de charger un seul pixel 4 bit). Ce qui veut dire plus concrètement qu’au moindre changement d'apparence de l’un des 16 sprites, même un simple flipping, il faut charger l'équivalent de 256 octets dans le chip graphique (qui est une tâche assez lourde pour un simple Z80 sans DMA) alors que les autres machines, pour la même action, se contentent la plupart du temps de modifier juste la valeur du pointeur vers le set de tuile en RAM et donc charger seulement un octet voir 2, ou modifier un attribut. Ça alourdit beaucoup toute la dynamique des sprites.
La GX-4000 permet aussi de faire du multiplexage software comme sur C64 mais la tâche est beaucoup trop lourde à cause de ce chargement des patterns... sauf si on se contente de multiplexer seulement les coordonnées de sprite pour dupliquer un objet sur l'écran (qui donc ne change pas d'apparence), par exemple des bullets. Dans ce cas la GX-4000 fait ça très bien avec des interruptions au raster et permet d'augmenter artificiellement le nombre de sprites de façon software quand le jeu s'y prête. Donc c'est un paradigme très différent. Une gymnastique qu’il faut apprivoiser pour en tirer le meilleur. Mais forcément, quand précédemment (sur CPC) tu avais juste des sprites software qu’il fallait construire au CPU, avoir cette alternatives est malgré tout un grand confort. Mais dans le contexte de son époque (1990) c’est quand même un pipeline de sprite hardware très archaïque.

Au moins le nombre de pixel-sprite par scanline et le pixel-coverage sont bon comparé au reste, équivalent aux consoles 16 bit, grâce au fait justement que toutes les patterns de sprites sont dans un cache interne au chip graphique et pas dans un set de tuile en RAM, c'est l'avantage de cette situation (ils avaient sans doute pas la bande passante pour mettre un tileset sprite en RAM). Une seule palette 15 couleur à disposition des sprites, similaire donc à la situation sur la Master System, mais avec une précision de couleur RGB 12 bit (4096 nuances) bien superieur, plutot similaire à la GameGear. J'ai pris 2 résolutions comme référence. Le 160x240 et le 320x240 (j'ai exclu le mode hires 640x pas très pertinent et adapté aux jeux et j'ai considéré que le 240 lignes était relativement accessible sur GX-4000) mais les jeux vont très majoritairement utilisé le lowres 160x comme sur C64. Les sprites ont des attributs de magnificence qui permettent de choisir la résolution des sprites indépendamment du background (ou pour s'aligner sur celle du background). Tout ça explique qu'il y a parfois des valeurs doubles dans les cases du tableau selon les choix de résolution.

PC-Engine

Les chips graphiques de la PCE ne sont pas des chips NEC mais bien des chips de conception Hudson Soft comme les autres chips de la console. Ca peut surprendre mais Hudson Soft a toujours eu un pied dans le hardware et force est de reconnaître qu'ils ont fait du très bon boulot. La PCE n'est pas vraiment une console 16 bit mais c'est elle qui va quand même ouvrir le bal des 16 bit. Et c'est flagrant quand on regarde du côté des sprites hardwares justement puisque la principale évolution apporté par les consoles 16 bit dans ce domaine sera un gros boost sur le nombre de pixel-sprite par scanline et donc sur le scanline coverage qui était le point faible des consoles 8 bit. Un boost x4 afin de viser les 100% de scanline coverage. La PCE va donc être la première console à viser cette objectif des 100% pour se rapprocher des machines d'arcade qui se distinguent particulièrement sur ce critère en général. On voit bien dans le tableau la rupture que la PCE propose sur ce point en comparaison de la NES et SMS malgré une liste de sprites qui reste similaire (64 sprites).

Mais l'autre point fort de la PCE va être le nombre de palettes disponibles. Les sprites PCE sont en 15 couleurs, dans la continuité du standard 4 bpp initié par la SMS, sauf que sur SMS on ne pouvait se construire qu'une seule palette 15 couleurs à partager avec tous les sprites. La PCE va offrir la possibilité de créer 16 palettes différentes. Quasiment chaque sprite peut donc avoir sa propre palette sur mesure et en piochant dans un nuancier codé sur 9 bit (512 nuances). Plus besoin de se faire des torsions neuronales impossibles pour construire une palette qui conviendra à tous les sprites à la fois. 16 palettes pour les sprites, c'est beaucoup plus que sur MD et même plus que sur SNES. C'est un confort certain pour les graphistes. La multiplication des palettes c'est aussi quelque chose que l'arcade pousse à l'extrême (souvent bien au-delà des besoins), donc là aussi on sent une volonté de singer un peu l'arcade. La PCE doit cet avantage non négligeable à son second chip vidéo dans lequel ont été délocalisés les palettes (il y en a autant dédiés aux backgrounds).

Un autre élément symptomatique du passage aux consoles 16 bit, et qu'on va retrouver ici sur la petite PCE, c'est la multiplication des formats de sprite avec des tailles plus imposantes (sur PCE ca va de 16x16 à 32x64) et qu'on peut choisir individuellement et donc mixer dans une même scène ou un même metasprite. Il ne s'agit pas cette fois d'imposer un format à toute la liste de sprite comme sur les consoles 8 bit. La seule lacune de la PCE sur ce point c'est de ne pas proposer de petit format de sprite 8x8. C'est un peu handicapant quand il s'agit par exemple d'afficher des bullets. Devoir utiliser des sprites 16x16 pour des bullets de quelques pixels ça revient à surcharger inutilement les scanlines (dans en sprite, les pixels invisibles / transparents sont tout autant une charge que les autres). Un autre élément qui peut manquer sur PCE est de ne pas pouvoir gérer les priorités entre sprite et background au niveau des tuiles de background, ca se fait uniquement au niveau des sprites comme sur NES (ça fait partie des quelques similitudes entre NES et PCE).

Le dernier point à évoquer pour mieux comprendre le tableau sont les résolutions. La PCE fait partie des machines qui proposent différents modes vidéo (et qui sont chacun très customisables). Pour le tableau je n'ai pris en référence que 2 résolutions. Le 256x240 (mode 5.37mhz qui représente 95% du catalogue) et le 352x240 (mode 7.16mhz, 5% du catalogue). Ca explique les doubles chiffres dans le tableau. Les chiffres les plus élevés du tableau représentent le mode 5.37mhz donc le plus standard sur PCE. J'ai exclu le mode hires 10.74mhz (512x) qui n'est pas vraiment utilisé (cf. Sherlock CD) ni très pertinent/adapté aux jeux contrairement au mode 7.16mhz qui, même si relativement marginal, est réellement utilisé dans des jeux d'actions et dans des jeux parfois cultes (notamment les jeux IREM comme R-Type ou Ninja Spirit).

Megadrive

La Megadrive fait partie de ces nouvelles machines qui visent un scanline et screen coverage de 100%. C'est à dire proposer un layer sprite qui peut couvrir l'intégralité de l'écran tel un layer background (on peut donc afficher un metasprite qui fait la taille de l'écran). La PC-Engine avait initié cela et la Megadrive confirme la tendance de cette nouvelle génération de console. L'originalité de la Megadrive c'est d'avoir des performances en sprite qui se scale en fonction de la résolution. En effet, la Megadrive propose 2 modes vidéo. Un mode 256x240 qui depuis le TMS9918 est devenu un peu le standard des consoles et qui ici permet aussi une forme de rétrocompatibilité avec la SMS en plus de facilité le multiplateforme avec la PCE et SNES qui utilise aussi principalement ce mode. Et un mode 320x240 qui est le vrai mode natif de la MD utilisé dans 90% des jeux. L'originalité vient des performances de sprite qui augmentent quand on utilise cette résolution plus élevée. La taille de la liste passe de 64 à 80 sprites, le nombre de sprites par scanline passe de 16 à 20 etc... Ca explique pourquoi le scanline et screen coverage reste à 100% même avec cette résolution supérieure contrairement par exemple à la PC-Engine qui voit son coverage régresser quand on utilise les résolutions supérieures. A l'inverse on a des valeurs doubles dans le tableau pour les autres critères comme les pixel-sprite par scanline et par screen. Donc sur MD il y a vraiment tout intérêt à utiliser ce mode 320x240 (en général les jeux se contentent de 224 lignes comme sur les autres consoles 16 bit). On peut même utiliser l'entrelacé (split screen Sonic 2) mais c'est anecdotique.

L'autre point notable des sprites MD sont les choix de format. Non seulement on peut choisir un format individuellement pour chaque sprite (contrairement à la SNES) mais la gamme de format proposée est particulièrement optimale. On est libre de choisir toutes les combinaisons possibles entre 8x8 (qui manque à la PCE) et 32x32, par palier de 8 pixels sur l'horizontale et/ou la verticale soit 16 formats de sprite différents au total. C'est à dire qu'on peut avoir des sprites 8x32 ou 32x16 mais aussi du 24x24 ou 24x8 etc… C'est beaucoup plus intéressant d'avoir toutes ces variantes a 24 pixels et ces petits formats de sprites pour optimiser au mieux la construction des metasprites et donc l'overflow (ainsi que l'espace VRAM), que d'avoir des formats 64x64 par exemple. Au Delà de 32x32 c'est pas très utile dans le contexte de contrainte et de performance des consoles 16 bits. Donc selon moi la MD propose la gamme de choix de formats de sprite la plus optimal des consoles 16 bit avec un max de liberté et de contrôle. Une liste de sprite simple à construire avec des commodités originales comme les link entre sprite pour contourner l'ordre de la liste et donc l'ordre des priorités. La liste de sprite MD autorise même des modifications au raster (contrairement à la PCE et SNES) pour faire du multiplexage software et donc dépasser la limite théorique de 80 sprites par screen (par exemple le split screen de Sonic 2 encore une fois). La Megadrive est aussi celle qui va introduire les fonctions DMA sur console pour accélérer les transferts de données entre mémoire et qui donc va aider aux mises à jour fréquentes de la VRAM notamment pour simplifier l'animation des sprites. Son DMA est même un peu plus rapide que celui de la SNES et aurait pu l'être encore bien plus s'il n'était pas artificiellement bridé par un bus 8bit du côté VRAM. La MD a failli être un monstre dans ce domaine.

Donc tout est vraiment au top pour 1988… si ce n'est le défaut génétique de la MD, toujours le même, qui impacte donc aussi les sprites, et qui est le nombre de palettes disponibles. Dans le paragraphe sur la PCE j'évoquais la possibilité de pouvoir construire 16 palettes différentes à l'usage exclusif des sprites. La Megadrive ne propose que 4 palettes (donc chaque sprite est associé à l'une de ces 4 palettes) et qui malheureusement ne sont pas dédiées aux sprites mais partagées aussi avec les layers background. Ca veut dire au mieux 2 à 3 palettes pour les sprites donc le chiffre de 60 couleurs cumulées dans le tableau est particulièrement surestimé car les sprites ont peu de chance de pouvoir s'accaparer toutes les palettes pour eux. Si peu de palette, relativement à une PCE ou SNES (ou à l'arcade), implique qu'il va falloir construire des palettes multi-usage. Pour chaque palette, le pixel-artiste doit donc faire des compromis et choisir 15 couleurs que devront se partager un large pool d'ennemis différents par exemple. Ça ne laisse donc pas autant de liberté qu'on voudrait dans le choix des couleurs. C'est pourtant bien une évolution par rapport à la SMS mais la PCE, sortie un an avant, a brisé ce sentiment et laisse plutôt un goût amer de contrainte pour les graphistes (c'est tout du moins le commentaire qu'on peut faire apres-coup. Sur le moment les graphistes devaient sans doute voir uniquement le côté positif de tout ce qu'apporterait la MD comme évolution). Sega tenait à tout intégré dans un seul chip graphique (si on excepte la partie production du signal vidéo). Il fallait donc savoir s'arrêter à temps pour ne pas faire exploser le prix du chip, là où la PCE et la SNES ont succombé à la fragmentation en 2 chips qui laisse un peu plus de marge. Les couleurs RGB des palettes sont codé sur 9 bit

comme sur PCE (512 nuances) sauf qu'en réalité la Megadrive a une output 12 bit exploitable par les fonctions shadow / highlight qui sont des features de FX pas très simple à bien exploiter (et donc peu utilisé) mais permettent d'étendre théoriquement la quantité total de nuance jusqu'à 1407 (sans parler de tricks encore plus tordus et marginals pour exploiter encore un peu plus la précision 12bit de l'output mais ca ne concerne pas vraiment les jeux).

SuperGrafX

En observant le tableau vous devez constater par vous même la corrélation évidente avec les chiffres de la PC-Engine. En effet la partie graphique de la SGX consiste simplement à prendre 2 chips graphiques PCE et à les monter en SLI (c'est à dire en parallèle avec le vocabulaire Nvidia) avec donc chacun leur VRAM comme du SLI (+ un 3ème chip, nouveau celui ci, qui a pour fonction de fusionner le flux des 2 autres). Donc la partie graphique de la SuperGrafx c'est littéralement 2 PC-Engine en parallèle (on retrouvera aussi cette configuration dans la PC-FX). Les chiffres des sprites sont donc tout simplement ceux de la PCE x2.
Il y a quand même un bémol. Ces chiffres théoriques sont à l'avantage de SGX en surestimant un peu la réalité car cette parallélisation implique une gymnastique un peu complexe si on veut vraiment exploiter au mieux les ressources. Il faut répartir le plus intelligemment possible les sprites entre les 2 chips graphiques pour éviter l'overflow par scanline propre à chaque chip et aussi répartir intelligemment les tuiles entre les 2 set de tuile et les 2 VRAM. Sans parler aussi des contraintes supplémentaires sur la gestion des priorités. C'est pas si simple et ça ne peut jamais être totalement optimal. Pour ces raisons, les performances effectives seront forcément inférieures à ce qu'elles seraient si ces chiffres venaient d'un unique chip graphique et une unique VRAM. Et puis faut piloter tout ça avec le même CPU 8 bit que la PCE qui doit toujours s'occuper aussi de l'audio. Heureusement il est quand même très véloce pour un CPU 8 bit, plus que celui de la SNES, mais on devine que ça devient quand même pas simple.
Une fois qu'on a dit ca, on peut dire que la proposition est quand même impressionnante car la PCE n'était déjà pas vraiment à la traîne sur le plan des sprites face aux consoles 16 bit (son handicap était plutôt d'avoir qu'un seul layer background) donc un x2 sur la partie sprite permet à la SGX de dépasser les capacités de la MD et de la SNES, même en prenant en considération les contraintes particulières de la configuration en SLI. Malheureusement on en verra pas grand-chose étant donné la taille famélique du catalogue. 1941 et Ghouls'n Ghost en font quand même bon usage. Ce surplus de sprite aurait pu notamment permettre à la SGX d'exploiter plus souvent le mode vidéo 7.16mhz et rendre le 352x240 majoritaire sur cette machine. Ça aurait permis aux jeux SGX de se démarquer de la SNES et la faible résolution de ses jeux.

SNES

Le chipset graphique de la SNES est physiquement composé de 2 chips (avant qu'ils ne fusionnent plus tard). Difficile de trouver par quel angle aborder la question des sprites SNES. A l'image du reste de la machine, il y a des choses intéressantes mais entachées par d'autres un peu bancales.
Je vais donc commencer par le chiffre le plus frappant, sa liste de 128 sprites, plus grosse que la concurrence. A priori c'est positif mais c'est plus complexe que ça. D'abord la taille de la liste de sprite n'est pas une mesure de performance d'affichage. Pour ça il faut plutôt regarder du côté des chiffres de pixel-sprite par scanline et par frame ou les chiffres de coverage. Sur ce point la SNES est comparable aux PCE et MD, c'est à dire qu'elle fait partie de cette catégorie de machine qui est tout juste en capacité de couvrir l'intégralité de son écran en sprite mais sans overdraw (ou juste un petit peu). Donc même si la SNES supporte des dimensions de sprite jusqu'a 64x64 pixels, elle serait bien incapable d'afficher 128 sprites 64x64 ou même 32x32 (ni 32x16 si le format existait). Même avec une répartition optimale des sprites sur la surface de l'écran, ça dépasse ses capacités d'affichage. Une grosse liste de sprites, c'est avant tout une commodité, pas une mesure de performance. C'est potentiellement utile si on a besoin d'afficher plein de petits sprites (par exemple plein de bullets ou d'effet de particules) ou pour optimiser au mieux les metasprites (décomposé un personnage en plein de petit sprite qui épousent au mieux la forme de celui ci pour optimiser l'overflow et la VRAM).
Donc c'est plutôt une bonne chose et tout de même un effort technique notable puisque ca demande au hardware de parcourir cette longue liste à chaque scanline. Sauf que, déjà, on peut se demander si les faibles ressources CPU de la SNES sont bien adaptées à construire et gérer une aussi grosse liste de sprites. Mais

surtout, la plupart des faiblesses concernant les sprites SNES vont venir de l'implémentation de cette grosse liste de sprite dans une mémoire cache interne trop petite. En effet, pour faire tenir cette liste de 128 sprites dans le petit cache interne de 544 octets, il a fallu faire des compromis. La SNES se retrouve ainsi avec des attributs de sprite codés sur 8 bit (sur PCE et MD les attributs sont plutôt dans des formats 16bit) qui, forcement, pose certains problèmes. Certaines features sont impactées comme le choix de format des sprites. La SNES propose seulement 4 formats de sprites: 8x8, 16x16, 32x32 et 64x64, donc très peu comparé à la MD (qui en propose 16) avec même de l'inutile (64x64). Et en plus de cela, on ne peut pas choisir individuellement le format de chaque sprite. Il faut d'abord choisir 2 formats (parmi les 4), qui seront imposés à toute la liste, et ensuite chaque sprite ne peut choisir qu'entre ces 2 formats. Donc des contraintes qui au final réduisent les possibilités d'optimisation des metasprites et contrebalance l'avantage d'avoir une grosse liste (à moins de choisir le combo de format 8x8 + 16x16 mais qui limite la capacité de couverture). Ça rend caduc aussi l'usage du format 64x64 trop peu utile pour justifier de gâcher l'un des 2 seuls choix. D'autant plus que l'attribut de coordonnée Y des sprites est bridé à 8bit (toujours pour les même raisons) ce qui donne un espace de coordonnée des sprites très étriqué verticalement (256 pixels), trop pour l'usage de sprite 64x64 qui pose problème si on veut les faire entrer ou sortir verticalement de l'écran. Et que dire aussi des coordonnées X des sprites qui, ne pouvant pas se contenter de 8bit, se sont vu affublée d'un 9ème bit mais isolé dans une autre table à part, tout comme le 9ème bit de la valeur qui sert à pointer les tuiles des sprites. Tout ça donne une liste de sprite fragmentée et complexe à construire qui demande encore plus de ressources CPU alors même que c'est le point faible de la SNES. On peut ajouter aussi que ce pointeur de tuiles bridé à seulement 9 bit signifie aussi que le tileset des sprites est limité à 512 tuiles et 16 Ko à un instant T (un quart de la VRAM) alors que sur MD ou PCE l'espace du tileset des sprites couvre l'intégralité de la VRAM (64ko). C'est potentiellement une contrainte supplémentaire dans certaines situations (ça veut dire par exemple que sur SNES on ne peut pas afficher un metasprite qui fait la taille de l'écran et qui est composé de tuiles uniques, sans redondance ou symétrie, contrairement à la MD et la PCE. Ce n'est pas un exemple d'usage très réaliste mais ca permet de quantifier plus concrètement cette contrainte).
C'est pas simple d'évoquer ce sujet complexe mais pour résumer il y a une forte contrainte sur le taille du cache interne dédié à la liste de sprite qui a imposé plein de petites contraintes qui n'existe pas sur MD et qu'on aurait aimé ne pas avoir en 1990. Mais rien de très grave non plus. Un truc plutôt cool c'est le clipping automatique des sprites hors champ. Un sprite qui est hors champ sur le côté de l'écran ne va pas peser sur le budget en pixel-sprite de la scanline, pas besoin de le dégager manuellement. Et puis pouvoir afficher 32 sprites 8x8 sur la même scanline ça peut être sympa pour du danmaku (à condition d'avoir les ressources CPU ^^). Du côté des palettes, la SNES s'en sort bien avec 8 palettes à disposition des sprites. C'est moins que sur PCE mais c'est suffisant pour être confortable et pour faire mourir d'envie la MD. Cela combiné à une précision de couleur élevé de 15 bit (32768 nuances), bien plus que sur PCE et MD (qui sont plutôt 9 bit soit 512 nuances, même si sur MD c'est un peu plus complexe avec son output 12bit), plutôt comparable aux systèmes d'arcade (ou NeoGeo) ce qui offre toute de même un certain confort dans le pixel art. Un très bon compromis et équilibre pour le coup. On peut même avoir de la transparence sur les sprites. On ne choisit pas le degré de transparence (figé à 50%) et on ne peut pas avoir de transparence entre sprites (seulement relatif au background) mais c'est quand même complètement nouveau. Quelque chose qui n'existe pas même sur NeoGeo ou CPS.
J'ai retenu uniquement la résolution 256x240 qui est le standard de tous les jeux SNES (c'est même plutôt le 256x224 qui domine largement), la même résolution que la NES qui, pour 1990, commence à être vraiment datée, mais c'est un standard qui a su s'imposer depuis le TMS9918 et pérenniser, sans doute parce que sortir des limites de 256 pixels c'est aussi sortir des commodités du format 8bit, ca demande un petit effort. Une résolution qui permet à la SNES d'obtenir plus facilement un coverage de 100% que la MD. Le mode hires 512x sur SNES n'est pas tout à fait un mode hires, c'est plus un trick hardware interne sur les layers background (d'ailleurs il ne s'applique pas sur les sprites, ni même sur le 3ème layer background ou le scrolling). Un mode qui n'est pas très pertinent pour les jeux et pas vraiment utilisé (le seul véritable exemple ingame est RPM Racing qui est aussi le seul à utiliser l'entrelacé ingame il me semble) donc je l'ai exclus des chiffres au même titre que le mode hires de la PCE / SGX ou celui de la GX-4000 et MSX 2 car pas représentatif.

Neo-Geo

Comment traiter la question des sprites hardwares sans parler de la reine du domaine dans le salon. Je vais probablement rien ne vous apprendre en vous disant que les performances exceptionnelles de la NeoGeo dans ce domaine s'explique en grande partie par un choix de design qui consiste à faire l'impasse sur les layers backgrounds pour les remplacer par des sprites afin de simplifier au maximum le chipset graphique et tout misé sur la force brute. Ca fait un peu écho à la Super Cassette Vision dont je parle aussi plus haut ou au Board Y de Sega qui bien avant (début 1988) poussait déjà l'idée plus loin en performance (mais pas avec des sprites au raster comme la NeoGeo). SNK le fait avec une certaine parcimonie et volonté de trouver un compromis performance/coût raisonnable afin de pouvoir submerger le marché arcade de machine à coût maîtrisé en s'appuyant notamment sur les bases hardware de ADK utilisé pour Time Soldiers. A la fois un monstre dans le salon et une machine très modeste pour un système d'arcade de 1990. Mais bien suffisant au final pour couvrir les besoins en jeux 2D classiques tant qu'on ne rechigne pas sur la taille des ROM ce que ne va pas faire SNK.

En ayant plus qu'un seul type d'objet graphique à manipuler (les sprites) ça donne effectivement une machine très simple. Les chipsets graphiques d'une Megadrive ou d'une SNES sont plus complexes et plus riches en features que celui d'une NeoGeo. Ce choix permet à la NeoGeo de consacrer 100% de ses ressources graphiques à l'affichage des sprites (là ou les consoles traditionnelles concentrent le traitement des sprites sur le court moment du Hblank entre le balayage de 2 scanlines). Cela grâce à un double line-buffer et un gros bus qui va piocher directement les tuiles graphiques des sprites dans la ROM de la cartouche. Ca permet à la fois de grosse performance mais aussi d'avoir constamment un accès direct et total à un énorme set de tuile qui peut faire jusqu'à plusieurs dizaines de Mo (jusqu'à 1 million de tuiles 16x16 soit 128 Mo, mais réduit à 4 Mo de buffer sur NeoGeoCD), là ou sur consoles il faut se contenter d'un buffer de tuile temporaire en VRAM limité à quelques dizaine de Ko à l'instant T. C'est la combinaison des 2 qui fait la force de la NeoGeo, performance et quantité.

J'ai dit que la NeoGeo manipule un seul type d'objet graphique, les sprites, mais ce n'est pas tout à fait vrai. D'abord ils ont quand même fait une exception à cette philosophie en ajoutant un "fix layer" qui sert à afficher le HUD sans avoir à gâcher des sprites. Et puis je parle de sprite mais en réalité le type d'objet graphique que la NeoGeo manipule est plutôt une sorte d'objet intermédiaire entre le sprite et le background. Des sortes de background 1D sous la forme de bande verticale. C'est à dire que ces objets sont des assemblages verticaux de tuiles 16x16 pixels (jusqu'à 32 tuiles) qui donne donc une gamme de format qui s'étend de 16x16 à 16x512. La liste de tuiles qui permet de construire ces bandes verticales s'apparente en quelque sorte à des tilemap de background mais sur une seule dimension (verticale). Ça reste plus simple tout de même d'appeler ces objets graphiques des sprites car c'est quand même ce à quoi ça ressemble le plus. Mais on comprend bien alors que les backgrounds vont être constitués d'un assemblage de grandes bandes verticales misent côte à côte. On peut linker les bandes entre elles pour piloter tout un bloc de bandes verticales avec un seul set d'attribut et former un background. Mais un scrolling reste quand même une tâche plus complexe qu'avec un vrai layer background classique tel que sur les autres machines.

Les chiffres du tableaux sont impressionnant avec une grosse capacité de pixel-sprite par frame et un screen coverage de 480%. Mais ce sont des chiffres nécessaires en l'absence de layer background. Il faut bien que les sprites NeoGeo soient capables de couvrir plusieurs couches d'écran pour compenser cela et construire le décors avec. Mais ça permet en quelque sorte à la NeoGeo de répartir ses ressources graphiques de façon optimale entre "sprite" et "background" selon les besoins spécifiques du jeu plutôt que d'avoir des ressources fixes et imposées pour chaque domaine. Reste que manipuler et construire une liste de 380 sprites doit tout de même être assez coûteux en ressources CPU. Il faut le CPU qui va avec.

Malgré la simplicité de son chipset, la NeoGeo propose quelques features. Une fonction auto-anim pour simplifier l'usage de petites animations dans le décor mais qui laisse peu de liberté car peu de paramètre. Et surtout la fonction de shrinking assez emblématique de la machine. Je vais sans doute rien vous apprendre non plus en vous disant que la NeoGeo n'a pas de fonction de zoom (et encore moins de rotation), elle a une fonction "shrinking" de rétrécissement (assez semblable à celui du System16B de Sega) qui consiste à utiliser un sprite nativement très gros pour le réduire à l'écran grâce à cette fonction et permettre ainsi une possibilité de "zoom" ultérieur qui consiste juste à restaurer sa dimension native sans shrinking (d'où l'absence de pixellisation lors des "zoom" qui n'en sont pas). La NeoGeo pouvant se permettre de manipuler de très gros sprites, cette approche alternative n'est pas un problème. Attention tout de même car un sprite shrinké consomme toujours autant de ressource que s'il était dans sa taille maximum donc il ne faut pas en abuser

non plus. Cette fonction de shrinking est un mélange de hardware (fonction câblé dans le pipeline sprite pour le shrinking horizontal + une table de 64 ko de valeurs précalculés dans une ROM dédié pour le shrinking vertical) avec une partie software à la charge du programmeur pour répartir le shrinking entre les différents sprites qui composent un personnage ou background.
Les pixel-sprite NeoGeo restent des pixels au format 4 bpp comme les autres consoles depuis la SMS mais avec une précision de couleur RGB 16 bit comparable aux autres systèmes d'arcade. En ce qui concerne les palettes, on est vraiment dans le monde extrême de l'arcade puisque cette fois il est question de 256 palettes disponibles (on s'éloigne vraiment de la contrainte des 4 palettes Megadrive). Autant dire que chaque objet, chaque personnage, chaque ennemi, chaque élément de décors, peut avoir sa propre palette construite sur mesure sans devoir faire de compromis. Une résolution classique de 320x224 mais avec un dotclok plutôt faible de 6 mhz qui implique que le 320x de la NeoGeo a une densité de pixel sensiblement plus faible que le 320x de la Megadrive (dotclock 6.71 mhz). Branché sur un même écran, l'image MD sera donc mieux résolue que l'image NeoGeo. L'autre conséquence c'est que l'image NeoGeo déborde largement dans l'overscan. Ce n'est pas vraiment problématique en arcade avec des moniteurs qui sont réglés sur mesure mais avec une AVS sur une TV standard on perd souvent un bout d'image sur les côtés si bien que certains jeux en profite pour se restreindre à une résolution 304x224 en cachant 8 pixels à gauche et à droite à l'aide du fixe layer (il n'y a pas de vrai mode 304x, c'est juste un trick).

Sega System 16
Un système d'arcade plutôt vieux (1986) et déjà relativement performant grâce à la combinaison d'un bus dédié uniquement aux tuiles graphiques des sprites (indépendamment de celui dédié aux backgrounds) et d'un double line-buffer qui permet de diluer le traitement des sprites sur toute la durée de la scanline comme sur NeoGeo. Ainsi malgré un débit de pixel-sprite plutôt modeste (2 pixel-sprite par dotclock soit 2 fois plus lent que sur les consoles 16 bit) le System16 obtient des performances bien supérieures aux consoles 16 bit qui sont contraintes de concentrer le chargement de tous les pixel-sprites sur la très courte période du Hblank entre 2 scanlines. Cela permet donc au System16 d'obtenir de bons chiffres de coverage (sachant qu'il y a aussi 2 layers background en complément contrairement à la NeoGeo). D'autant plus qu'il y a d'autres commodités pour optimiser cet aspect. En effet les sprites sur le System16 ne sont pas des assemblages de tuiles comme sur les autres systèmes. Ils sont plutôt comme des bitmap avec donc une grande liberté sur le choix des dimensions. On peut choisir à peu près tous les formats entre 7x1 pixels et 511x256 (seule contrainte, une granularité minimale de 4 pixels pour la dimension horizontale pour matcher avec le bus 16 bit) ce qui permet d'optimiser la taille des sprites selon les besoins. Et mieux que ça, on peut placer une commande de fin de ligne dans le flux de pixel pour stopper le rendu d'une ligne de sprite au plus tôt. Pas besoin de tracer toute la ligne du bitmap s'il ne reste que des pixels invisibles / transparents à tracer. De ce fait, c'est comme si on pouvait faire varier la largeur du sprite pour chacune des lignes qui le compose afin d'épouser au plus près la forme du sprite et ainsi optimiser au mieux l'overflow de la scanline et donc le coverage effectif. La commande de fin de ligne est simplement l'une des couleurs de la palette. Pour toutes les machines, la première couleur de la palette (couleur 0) est en général réservée aux pixels transparents nécessaires à tout sprite pour lui donner les contours que l'on souhaite. C'est bien le cas aussi sur System16 sauf que sur celui-ci la dernière couleur de la palette (couleur 15) est aussi réservée pour indiquer cette fin de ligne prématurée. La contrepartie est donc une réduction de la palette des sprites à 14 couleurs au lieu des standard 15 couleurs des autres machines 4 bpp.
Le System16B propose aussi une fonction de shrinking de sprite comme sur NeoGeo. A la différence que la réduction de taille du sprite ne va pas au-delà de 50% alors que sur NeoGeo le shrinking couvre la totalité de la gamme jusqu'à 0%. Mais avoir un shrinking bridé à 50% n'est pas un mauvais compromis car un shrinking trop élevé crée divers artefact visuel à cause de problèmes d'échantillonnages donc il est de toute façon préférable d'appliquer une méthode de type "mip mapping" tel que pour les textures en 3D. C'est à dire utiliser des variantes de la texture ou du sprite original mais dans des dimensions ½, ¼, ⅛… et utilisé le shrinking uniquement pour transiter entre ces versions intermédiaire qui, elles, sont propre, sans problème d'échantillonnage. Avec cette méthode, pas besoin d'avoir un shrinking inférieur à 50%, et au pire ca consomme seulement +30% d'espace pour avoir toutes les variantes intermédiaires du sprite. On voit beaucoup de shrinking dans Altered Beast par exemple (la mort des loups, la mort des zombies, les nuages

de fumée à la mort des boss, les boules de feu du boss du stage 4…). En général un usage basique qui se contente de la limite de 50% nativement supporté par la fonction.
Du côté des couleurs on retrouve la même précision de codage 16 bit que sur CPS ou NeoGeo (65 536 nuances) et un pool de 64 palettes à disposition des sprites (sans compter que les layers background ont leur propre pool de palettes) donc tout à fait suffisant pour être très confortable. Une résolution classique de 320x224 comme sur NeoGeo (avec un dotclock intermédiaire entre celui de la NeoGeo et de la MD).
Globalement le System16 n'est quand même pas très éloigné d'une NeoGeo quand on ajoute les layers background dans l'équation qui offre un certain confort. La différence va peut être se faire sur le format 3 bpp des pixels background sur System16 qui donc sont bridés localement à seulement 7 couleurs au lieu de 15.
Mais l'autre grosse différence c'est surtout la taille des jeux. Un grand classique du System16 comme Shinobi c'est seulement 1 Mo de ROM. On est loin des gros titres NeoGeo car ce n'est pas la même époque, la vie de la NeoGeo s'étalant sur une vaste période. Une période particulière où la qualité graphique est intimement liée à la quantité de ROM. Le System16 reste un système Sega assez modeste malgré tout. Une X board (Afterburner) ou Y Board (Galaxy Force) de Sega qui précède aussi le CPS1, et donc sensiblement plus vieille qu'une NeoGeo, sont de vrais monstres comparé à ces dernières qui ont des objectifs de coût plus raisonnables et pragmatiques.

Capcom CPS1
Le CPS est un cas particulièrement intéressant car un peu en marge. Toutes les autres machines de ce comparatif ont un traitement des sprites au raster. C'est à dire un traitement des sprites qui se fait au fil du balayage de l'écran (rasterisation) donc au tout dernier moment, juste avant que les pixels concernés soient affichés (ou avec une scanline d'avance). "Racing the beam", c'est une course avec le canon à électron qui nécessite de parcourir toute la liste de sprite à chaque scanline pour déterminer ce qui sera afficher a la prochaine scanline ou la suivante. Le CPS se détache de ce paradigme pour adopter une autre approche qui consiste à construire le layer sprite tout simplement dans un (double) frame-buffer. Plutôt que de faire la course avec le canon à électron, le CPS va donc accumuler les sprites un par un dans un buffer RAM qui fait la taille de l'écran. Il va pouvoir prendre tout le temps d'une frame pour accomplir cette tâche tranquillement et afficher le résultat à la frame suivante. A l'inverse, les layers background sont toujours traités au raster, à l'ancienne.
C'est donc une approche hybride qu'on peut comparer à celle de la X Board de Sega exploitée l'année d'avant (1987) ou d'une console Saturn. Mais en général cette approche est plutôt destinée à faire du scaling, rotation ou déformation de sprite hardware, voire du polygone. C'est un peu plus inattendu pour des sprites 2D parfaitement classique (il n'y a aucune fonction de zoom, rotation ou autre sur CPS) mais ça permet de diluer la charge de façon optimale. Les autres machines qui traitent les sprites au raster ont une capacité théorique de traitement bien plus élevé (qui s'apparenterait au chiffre de pixel-sprite par frame dans le tableau) mais en pratique elle ne vont en exploiter qu'une petite portion selon comment se répartissent les sprites sur les différentes scanline (rarement de façon optimal) pour profiter du multiplexage hardware. Le CPS se fiche de cette répartition des sprites à l'écran. Ils peuvent être tous sur la même scanline, ça ne change rien pour lui, la charge reste la même et le CPS fera toujours un usage optimal de ses ressources en sprite hardware. C'est cela qui explique à la fois le petit chiffre de pixel-sprite par frame qui est bien inférieur à celui de la NeoGeo, et même inférieur à celui de la Megadrive, et en même temps un chiffre de pixel-sprite par scanline qui explose toutes les autres machines. Pour les mêmes raisons on constate un scanline coverage très élevé mais un screen coverage très faible. On est donc sur 2 approches très différentes qui donnent des chiffres antagonistes aux autres machines et rendent les comparaisons difficiles.
Cette approche plus optimal dans l'usage effectif des ressources permet au CPS de diluer la charge et de se contenter d'une cadence de traitement des pixel-sprite assez faible (conditionné aussi par l'obligation de devoir passer par une RAM externe plus lente que les mémoires caches internes utilisé dans les machines au raster). Le CPS1 traite une seule tuile 16x16 par scanline (256 tuiles par frame) soit 1 seul pixel-sprite tous les 2 dotclock contre 4 pixel-sprite par dotclock pour les consoles 16 bit par exemple. Cette cadence de traitement faible permet aussi d'éviter d'avoir un stress élevé sur la ROM qui contient les patterns graphiques et donc de pouvoir se contenter de ROM de qualité modeste et ainsi pouvoir en mettre un maximum car la quantité de ROM reste un élément déterminant dans la qualité graphique des jeux de cette époque. Donc tout ça a du sens mais ça implique quand même un surcoût du côté de la RAM. Ce double buffer, destiné au layer sprite,

utilise 384 ko de RAM à lui tout seul, 3 fois plus que toutes les RAM combinées de la NeoGeo (en contrepartie ça simplifie un peu le chip graphique).
N'avoir aucune contrainte de sprite par scanline c'est un vrai soulagement car c'est en général ici que ça coince sur les autres machines qui nécessite parfois de se creuser la tête pour éviter certaine situation d'overflow. Mais si on regarde ça d'un point de vue plus global, les capacités en sprite du CPS1 ne sont pas très élevées, voire un peu décevantes. Une liste de 256 sprites avec des dimensions entre 16x16 et 256x256 ça fait plutôt rêver... sauf que comme je vous l'ai expliqué plus haut, la capacité de traitement de sprite du CPS est plutôt faible et donc au mieux le CPS peut afficher 256 sprites de 16x16 pixels ou un seul sprite de 256x256 pixels mais évidemment pas 256 sprites de 256x256 pixels. Finalement les jeux vont principalement utiliser des sprites 16x16 pour optimiser la construction des metasprites (personnages) afin de ne pas gâcher de ressource. Il n'y a finalement pas beaucoup d'intérêt à utiliser les formats de sprite plus imposants qui n'augmentent pas concrètement la capacité d'affichage voir en gâche. Le paradoxe est que le CPS1 n'est donc pas capable d'afficher un metasprite de la taille de l'écran (son max screen coverage est seulement de 76%), ce que peuvent pourtant faire les SNES, Megadrive ou même PC-Engine. Les jeux CPS1 flirtent souvent avec les limites, voire les dépassent (par exemple certaines scènes de Final Fight). Une disparition de sprite ou un clignotement dans un jeu CPS va donc plutôt venir d'un dépassement globale du nombre de sprites plutôt qu'une surcharge locale de la scanline comme sur les autres systèmes, mais ça reste rare a priori, les devs évitent au mieux ce genre de situation et 256 sprites 16x16 ca reste quand même pas mal. Donc encore une fois c'est une approche différente avec des atouts et inconvénients différents mais qui, au vu du résultat dans les jeux, semble être plutôt un bon compromis. Il ne faut pas négliger aussi que le CPS à 3 layers de background (qui sont tous full 4 bpp contrairement à la SNES par exemple) donc rien n'empêche par exemple d'utiliser l'un des 3 layers comme un très gros sprite (ce qui ajoute potentiellement +100% de screen coverage) si vraiment il y a besoin.
Il reste quand même un autre petit inconvénient à cette approche de construction du layer sprite dans un buffer RAM. L'affichage des sprites se fait à la frame suivante avec donc un retard d'une frame par rapport aux autres machines de ce comparatif qui le font au raster. Le CPS a donc nativement une frame d'input lag supplémentaire par rapport à une NeoGeo ou un System16 (pour rester dans la concurrence directe). C'est négligeable mais ça peut intéresser les puristes (vous pouvez même le constater vous même à l'aide du frame advance de Mame en comparant un Shinobi, Magician Lord et Ghouls'n Ghost). On peut aussi signaler que malgré l'utilisation d'un frame buffer, le CPS ne laisse aucun contrôle dessus et donc on ne peut pas profiter de celui-ci pour répartir la charge des sprites sur plusieurs frames (pour doubler la capacité d'affichage en faisant tourner le jeu à 30 fps au lieu de 60 fps par exemple) contrairement au FM Towns par exemple, qui utilise aussi un frame buffer pour son layer sprite hardware ce qui lui permet, à 30 fps, de dépasser la capacité en sprite du CPS. Donc le CPS reste un hardware calibré pour le 60 fps comme les machines aux raster qui composent le reste de ce comparatif.
En ce qui concerne les couleurs, on reste sur du pixel classique 4 bpp avec une précision de codage des couleurs RGB en 16 bit comparable à la concurrence directe. Il y a 32 palettes à disposition des sprites. C'est sensiblement moins que sur NeoGeo et même moins que sur System16 (on se rapproche même des 16 palettes de la PCE) mais ça reste suffisant si on ne fait pas n'importe quoi. C'est juste que les systèmes d'arcades sont souvent surdimensionnés sur ce point. Une grande proportion de leurs palette ne sont jamais utilisées dans les jeux. Le CPS se distingue plutôt par sa résolution. L'effort est notable avec une belle densité de pixel horizontal (384x224) qui permet d'avoir une finesse d'image supérieure aux consoles, NeoGeo ou System16. C'est un choix de résolution qui donne des pixels un peu anamorphique (contrairement au classique 320x) mais c'est intéressant de pousser un peu plus la finesse graphique (à condition d'avoir un écran de bonne qualité). On avait déjà pu voir la même proposition chez IREM et son M72 (R-Type par exemple). Par contre c'est un choix de résolution qui va compliquer un peu les adaptations sur consoles en obligeant souvent à adapter et interpréter les graphismes si on veut des proportions correctes.

Sharp X68000
Contrairement à ce qu'on peut entendre parfois, le hardware du X68000 est radicalement différent du CPS, surtout la partie graphique qui se compose à la fois d'une partie "micro" qui est le plus gros morceau et qui va être utilisée pour construire les layers background de façon software dans un framebuffer tel un Atari ST ou Amiga (et donc plutôt coûteuse en ressource CPU) avec 1 Mo de VRAM. Et une partie "console" au raster qui

prend la forme d'un chip custom appelé Cynthia qui ressemble beaucoup à un chip graphique de console 16 bit (en un peu plus simple), isolé de la partie graphique "micro" (même physiquement puisque pas sur la même PCB), avec sa propre VRAM de 32 ko. Évidemment c'est Cynthia qui va être responsable des sprites hardwares (de façon classique, au raster, contrairement au CPS) qui nous intéresse ici. Ces 2 parties graphiques du X68000, composé chacune de plusieurs layers, sont ensuite mixées en output.
Cynthia est aussi capable de prendre en charge 2 vrais layers background avec tilemap tel une Megadrive mais la très faible quantité de VRAM associé à Cynthia compromet cet usage. En effet avec seulement 32 Ko (2 fois moins que la VRAM d'une PC-Engine et 4 fois moins qu'une SGX) pour contenir à la fois les tilemaps, le set de tuile des backgrounds et le set de tuile des sprites, on est vite coincé et il faut faire un choix. De ce fait les jeux utilisent presque essentiellement Cynthia pour ses sprites hardware et laisse les background à la charge de la partie "micro" ce qui est un peu dommage et donne parfois l'impression que Cynthia est une pièce rapportée pas très bien intégrée. Même en réservant toute la VRAM pour les sprites, ça reste pas énorme et ça sera l'une des contraintes. D'autant qu'il n'y a même pas de DMA, comme on peut avoir sur MD et SNES, pour faciliter la mise à jour dynamique de la VRAM. Sur X68000 beaucoup de choses se font au CPU (et avec beaucoup de cycle de pénalité pour accéder à la mémoire de Cynthia qui est relativement isolé) mais pour un micro de 1987 ça reste tout de même des conditions exceptionnelles.
Cynthia offre toute même une liste confortable de 128 sprites hardwares et avec de très bonne performance en pixel-sprite par scanline, soit 512, qui sera son point fort (sans doute que l'objectif était d'avoir un scanline coverage de 100% même en hires 512x d'où ce chiffre confortable). Donc à priori des conditions supérieures aux consoles 16 bit (et très similaire à une SupergrafX, il y a beaucoup de similitudes entre Cynthia et une SGX) sauf qu'il y a un autre problème. Il n'y a aucun choix de format de sprite. Tous les sprites sont au format unique de 16x16 pixels. On est loin des multiples format proposés sur les consoles 16 bit. Et 16x16 c'est plutôt un petit format. Ainsi même en cumulant les 128 sprites, on ne peut pas couvrir une grande surface, d'où la faible valeur de pixel-sprite par frame et de screen coverage dans le tableau. On est loin de pouvoir afficher un metasprite de la taille de l'écran comme sur les consoles 16 bit. Si seulement il y avait eu au moins le choix entre 16x16 et 32x32 ca aurait suffit à régler ce problème de coverage. Mais quelque part ça donne une connexion avec le CPS qui, pour des raisons différentes expliquées dans le paragraphe le concernant, va aussi utilisé principalement des sprites au format 16x16.
Cette capacité limitée à couvrir l'écran de sprite on va la ressentir par exemple dans Final Fight qui devient un jeu 2 players versus 4 ennemis sur X68000 alors que la version originale sur CPS propose un gameplay 2 vs 10. En effet un personnage de Final Fight (metasprite) est composé en moyenne d'une vingtaine de sprites sur X68000 ou CPS. Cynthia atteint déjà ses limites ici (en plus du CPU car sur X68000 la partie graphique est une charge conséquente pour le CPU comparé aux consoles et systèmes arcade). Afterburner est même contraint de gérer les sprites (F14, ennemis, missiles, fumée…) de façon software, à la charge de la partie micro, donc au CPU dans un framebuffer (comme un Atari ST par exemple), pendant que Cynthia se limite à afficher les sprites qui habillent le sol qui défile sous l'avion. La quantité de pixel-sprite nécessaire dans un jeu de ce type est bien trop élevée pour Cynthia. L'usage de sprite software explique ainsi le faible framerate du jeu (15-20 fps, ce qui est vraiment pas mal dans ce contexte, le code est bien optimisé). En recours ultime on peut aussi utiliser un layer tilemap de Cynthia comme un gros sprite, mais on se retrouve alors confronté à nouveau à l'autre problème de Cynthia, la faible quantité de VRAM de 32Ko. Final Fight utilise quand même cette astuce parfois comme à l'intro du jeu. Les 6 bidons qui font barrage entre le joueur et sa bien-aimée en train de se faire kidnapper, ne sont pas des sprites comme c'est le cas sur la version CPS. 5 des 6 bidons sont incrustés dans un layer background de Cynthia (le 6ème est un vrai sprite pour que l'ennemi puisse le bousculer) autrement cette scène serait déjà trop chargée en sprite.
Cynthia manipule des pixels au format classique 4 bpp avec une précision de couleurs 16 bit comparable aux systèmes d'arcade et 16 palettes disponibles. Elles sont normalement à partager avec les layers background de Cynthia mais comme Cynthia est souvent utilisé uniquement pour ses sprites, on peut considérer que les 16 palettes sont disponibles pour les sprites (les layers du côté "micro" ont leurs propres palettes), donc équivalent à une PC-Engine ce qui est très confortable et proche de l'arcade et du CPS. Le X68000 offre une certaine liberté pour bidouiller divers modes vidéo en jouant sur quelques paramètres. J'ai donc fait un choix arbitraire de sélectionner une seule résolution de référence pour simplifier le tableau comparatif et j'ai choisie le 384x224 qui est la résolution des jeux CPS. En effet il se trouve qu'en bidouillant on peut reproduire un mode vidéo proche de celui du CPS (le dotclock ne sera pas le même, au mieux ca sera du 8.7mhz vs 8mhz sur CPS, donc un format de pixel et d'image un peu différent). Quand on pense aux X68000 on pense souvent

en premier lieu aux adaptations des jeux CPS (et au mythe de la filiation entre ces 2 machines) donc ça me paraissait pertinent de choisir cette résolution pour simplifier la comparaison avec le CPS. Et puis aussi parce que c'est une sorte de résolution intermédiaire entre les résolutions lowres comme le 256x224 utilisé dans certains jeux (Afterburner par exemple) et des résolutions hires en 512x (mais je n'ai pas vraiment d'exemple de jeux qui utiliseraient ce type de résolution autrement que de façon partiel pour fenêtrer du 384x par exemple) donc un bon compromis.