(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) **EP 1 738 349 B1**

(12) **FASCICULE DE BREVET EUROPEEN**

(45) Date de publication et mention de la délivrance du brevet:
29.06.2016 Bulletin 2016/26

(21) Numéro de dépôt: **05753728.4**

(22) Date de dépôt: **08.04.2005**

(51) Int Cl.:
G09G 5/42 *(2006.01)*

(86) Numéro de dépôt international:
PCT/FR2005/000857

(87) Numéro de publication internationale:
WO 2005/104086 (03.11.2005 Gazette 2005/44)

(54) PROCEDE ET SYSTEME DE CONSTRUCTION VOLATILE D'UNE IMAGE A AFFICHER SUR UN SYSTEME D'AFFICHAGE A PARTIR D'UNE PLURALITE D'OBJETS

VERFAHREN UND SYSTEM ZUR VOLATILEN ERSTELLUNG EINES AUF EINEM ANZEIGESYSTEM AUSGEBBAREN BILD AUS EINER VIELZAHL VON OBJEKTEN

METHOD AND SYSTEM FOR VOLATILELY BUILDING AN IMAGE DISPLACEABLE OF A DISPLAY SYSTEM FROM A PLURALITY OF OBJECTS

(84) Etats contractants désignés:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IS IT LI LT LU MC NL PL PT RO SE SI SK TR

(30) Priorité: **08.04.2004 FR 0403721**

(43) Date de publication de la demande:
03.01.2007 Bulletin 2007/01

(73) Titulaires:
- **Hauttecoeur, Philippe**
  91940 Les Ulis (FR)
- **Rostan, Hervé**
  78125 Mittainville (FR)

(72) Inventeurs:
- **Hauttecoeur, Philippe**
  91940 Les Ulis (FR)
- **Rostan, Hervé**
  78125 Mittainville (FR)

(74) Mandataire: **Pontet Allano & Associes**
Parc Les Algorithmes, Bâtiment Platon
CS 70003 Saint-Aubin
91192 Gif-sur-Yvette Cedex (FR)

(56) Documents cités:
EP-A- 0 860 810
EP-A- 0 955 625
GB-A- 1 503 362
US-A- 5 699 076
US-A- 5 745 095

EP 1 738 349 B1

Printed by Jouve, 75001 PARIS (FR)

Description

[0001] La présente invention se rapporte à un procédé pour construire une image à afficher sur un système d'affichage, tel qu'un écran à partir d'une pluralité d'objets. Un objet est un élément graphique qui peut être affiché à l'écran. Il peut être de type vidéo, bitmap ou vectoriel par exemple.

[0002] Si l'approche "objet" est une notion répandue au niveau applicatif, les mécanismes d'affichage actuels se contentent traditionnellement d'aller chercher l'image déjà construite quelque part en mémoire, dans une zone prédéfinie qui fait généralement la taille d'une trame vidéo, d'où son nom de "mémoire de trame" ou "frame buffer" en langue anglaise. Entre un module applicatif gérant des objets, et un module d'affichage exploitant une mémoire trame, un module de copie d'objets est communément utilisé permettant de récupérer un objet dans une zone quelconque et de venir l'écrire dans la mémoire trame.

[0003] Les systèmes graphiques et vidéo selon l'art antérieur utilisent des mémoires de trame dans lesquelles les images sont fabriquées à partir de primitives graphiques et vidéo et d'objets copiés dans ces zones. Lorsqu'il existe plusieurs plans mémoire, ceux-ci sont généralement recombinés au moment de l'affichage en utilisant des mécanismes de multiplexage, d'incrustation ("chroma-keying" en langue anglaise) ou de fusion par transparence ("alpha-blending" en langue anglaise). Cette approche est très gourmande en espace mémoire, surtout pour les systèmes désireux d'offrir une très bonne qualité visuelle, et qui requièrent alors de multiples plans mémoire. Par ailleurs, à chaque fois qu'un objet graphique est remplacé par un autre, supprimé ou déplacé, il y a reconstruction totale de la mémoire trame dans laquelle il se trouve, d'où une perte en fluidité.

[0004] Pour les systèmes standards, le dynamisme et la qualité visuelle que l'on veut apporter à une Interface Homme-Machine sont complètement dimensionnant pour le coeur processeur du système (matériel et/ou logiciel, "hardware" et/ou "software" en langue anglaise) et pour la quantité de mémoire. Ainsi :

- plus le processeur sera puissant, plus il pourra réaliser un grand nombre d'opérations lors de la préparation des plans images et gérer des modifications importantes de contenu d'une trame à la suivante, et plus il pourra manipuler un grand nombre de plans mémoire;
- plus il y a de plans mémoire, plus le nombre de niveaux graphiques est important, ce qui accroît les possibilités en terme de transparence complexe dans l'image finale.

[0005] La présente invention vise à remédier aux inconvénients précités en proposant un procédé de construction d'image dans lequel la mémoire utile à cette construction et la puissance utile du système sont réduites par rapport aux systèmes de l'art antérieur.

[0006] On atteint le but précité avec un procédé pour construire une image apte à être affichée sur un système d'affichage à partir d'une pluralité d'objets, comme définit par la revendication 1.

[0007] Contrairement aux systèmes de l'art antérieur, la présente invention permet d'afficher directement des objets sur un système d'affichage tel qu'un écran. L'objet est au coeur du mécanisme de construction de l'image. On dit que la construction de l'image est volatile puisque son mécanisme n'a pas recours à une mémoire trame et que l'image n'existe nulle par ailleurs que sur le système d'affichage. En d'autres termes, on construit l'image ligne par ligne en temps réel depuis les lieux où sont disséminés les objets jusqu'au système d'affichage. Alors que dans les systèmes de l'art antérieur, on construit d'abord l'image à partir des objets, on stocke cette image dans une mémoire trame, puis on affiche l'image déjà formée.

[0008] Par système d'affichage, on entend un système comprenant un ou plusieurs écrans synchronisés en ligne et en trame, et pilotés par un contrôleur unique générant des signaux de cadencement des écrans, ce contrôleur étant disposé dans l'accélérateur. Par exemple, du point de vue de l'accélérateur, deux écrans côte à côte peuvent être considérés comme un seul écran de définition double.

[0009] D'autre part, le système d'affichage peut comprendre une mémoire d'affichage apte à recevoir les pixels formant l'image. L'image est alors formée au sein de cette mémoire d'affichage plutôt que directement sur un écran.

[0010] Par ailleurs, lorsque les objets graphiques ou vidéo sont stockés en mémoires sous forme compressée, la présente invention présente un avantage certain par rapport aux systèmes standards puisqu'elle nécessite peu de mémoire (absence de mémoire trame), donc un coût de revient inférieur.

[0011] En outre, puisque l'image est construite directement, l'opération de préparation de la trame dans une mémoire trame est évitée, d'où une diminution de la bande-passante nécessaire au niveau de la mémoire. Ayant moins d'opérations à effectuer pour afficher une image à l'écran, le système tout entier peut être moins puissant, donc moins consommateur d'énergie, moins perturbant, ce qui est primordial pour un système embarqué par exemple. La présente invention permet donc de réaliser des images d'une très grande qualité visuelle à partir de composants faible puissance.

[0012] Puisque l'image que voit un observateur n'existe que sur l'écran et n'est pas stockée, l'image doit être reconstruite à chaque nouvelle trame vidéo. En conséquence, la réalisation de séquences d'images complexes avec des objets dynamiques ne nécessite aucune puissance supplémentaire par rapport à ce qui est nécessaire pour afficher toujours la même image. En revanche, les systèmes standards sont rarement capables de mettre du dynamisme dans leur Interface Homme-

Machine (IHM) ou bien le font au détriment de l'aspect temps réel puisque le nombre important d'opérations nécessaires à la préparation de l'image dans la mémoire trame peut dépasser la durée de la trame vidéo de l'écran. Pour un coprocesseur graphique standard, l'aspect dynamique d'une image (d'une IHM) est dimensionnant. Avec la présente invention, une image dynamique peut être obtenue sans re-dimensionnement puisque le nombre d'opérations à réaliser dans le temps trame sera le même. On entend par image dynamique, une image composée d'objets 2D pouvant évoluer au fil du temps, se déplacer, se transformer, changer d'état, ... Des modes de mise en oeuvre avantageux sont définis par les revendications dépendantes

[0013] Suivant un autre aspect de l'invention, il est proposé un système pour construire une image apte à être affichée sur un système d'affichage à partir d'une pluralité d'objets notamment stockés en mémoire vive, comme définit par la revendication 10.

[0014] Avantageusement, l'accélérateur matériel comprend les éléments suivants :

- un gestionnaire d'objets pour activer et caractériser des traitements liés aux objets,
- une mémoire de stockage des paramètres et variables de chaque objet,
- un module matériel de type DMA ("Direct Memory Accesss") en langue anglaise) contrôlé par l'accélérateur (et non par l'unité de traitement) apte à récupérer des données des objets depuis une mémoire,
- une mémoire tampon pour stocker temporairement les données provenant du module matériel DMA,
- un module de décompression et de conversion de données brutes provenant de la mémoire tampon en pixels,
- un multiplexeur pour sélectionner le pixel à afficher entre celui provenant du module de décompression et de conversion et celui provenant d'une source externe, à la condition que la source externe, l'accélérateur et l'écran soient synchronisés,
- un mélangeur pour mélanger le pixel provenant du multiplexeur et un pixel actuellement affiché en fonction de la transparence du pixel provenant du multiplexeur, et
- deux mémoires ligne assurant successivement à tour de rôle l'affichage sur la ligne courante du système d'affichage de pixels préalablement stockés, et le stockage de pixels provenant du mélangeur et qui sera affiché à la ligne suivante.

[0015] A la façon d'un OSD (On Screen Display), le présent système comprend des moyens pour insérer/incruster des informations graphiques dans un flux vidéo principal. A la différence de celui-ci, les informations insérées sont des objets hétérogènes et paramétriques.

[0016] D'autres avantages et caractéristiques de l'invention apparaîtront à l'examen de la description détaillée d'un mode de mise en oeuvre nullement limitatif, et des dessins annexés, sur lesquels :

- La figure 1 est une vue schématique simplifiée d'un processus de construction et d'affichage d'une image selon l'art antérieur;
- La figure 2 est une vue schématique simplifiée d'un processus de construction et d'affichage d'une image ligne par ligne en temps réel selon la présente invention;
- La figure 3 est un schéma illustrant deux zones temporelles d'une trame vidéo selon l'invention;
- La figure 4 est un schéma illustrant la façon dont deux mémoires ligne sont utilisées pour la construction et l'affichage d'image ligne par ligne selon l'invention; et
- La figure 5 est une vue schématique simplifiée d'un accélérateur matériel mettant en oeuvre le procédé selon la présente invention.

[0017] Sur la figure 1, on désire réaliser et afficher sur un écran 1 une image à partir d'une pluralité d'objets 2 contenus dans une mémoire vive 3. Pour ce faire, selon l'art antérieur, un module 4 de copie d'objets construit ladite image qui est ensuite stockée dans une mémoire trame 5. Cette dernière fait généralement la taille d'une trame vidéo. Un module d'affichage 6 se contente alors d'aller chercher l'image déjà construite dans la mémoire trame 5 et de l'afficher sur l'écran 1 pendant la durée d'une trame vidéo.

[0018] La présente invention met en oeuvre un procédé complètement différent. Sur la figure 2, on désire également afficher une image sur l'écran 1 à partir d'une pluralité d'objets 2 stockés en mémoire vive 3. Pour ce faire, on utilise un processeur matériel ou "hardware" en langue anglaise, autrement appelé accélérateur matériel graphique, pour construire à la volée et en temps réel ladite image à partir des objets 2. Cet accélérateur, notamment représenté sur la figure 5, réalise différentes opérations au cours de la trame vidéo et les répète à chaque nouvelle trame. Ainsi, durant une trame, pour chaque ligne active de la trame, le système selon l'invention construit en temps réel les pixels devant s'afficher sur cette ligne à partir des objets 2. En d'autres termes, l'accélérateur matériel comprend un module 7 pour construire les pixels d'une ligne courante de l'image à afficher, au moins une mémoire ligne 8 pour stocker temporairement les pixels ainsi construits, et un module d'affichage ligne 9 sur l'écran 1.

[0019] D'une façon générale, dans la trame vidéo qu'il génère ou sur laquelle il se synchronise et se verrouille, l'accélérateur matériel identifie alors, conformément à la figure 3, deux zones temporelles distinctes qui sont :

- la zone morte verticale (dite "vertical blanking interval", VBI)
- la zone active verticale (dite "vertical active interval", VAI)

[0020] Le mécanisme de construction volatile selon un example comparatif s'appuie sur ces intervalles de temps pour réaliser l'affichage d'une image composée d'objets à l'écran et stables le temps d'une trame. Macroscopiquement, on distingue le VBI et le VAI. On profite du VBI de la période trame pour réaliser toute la préparation nécessaire au bon déroulement de la construction volatile qui se déroulera dans le VAI par un processus ligne. Pour chaque ligne active de l'écran, la construction volatile consiste à remplir une mémoire de ligne avec les segments de ligne des objets qui sont actifs dans la ligne considérée. Le contenu de la mémoire de ligne est ensuite envoyé à l'écran au rythme de la fréquence pixel. Comme on le verra ci-dessous avec la figure 4, la présente invention permet à ce que lorsqu'on affiche une ligne (L-1), pendant ce temps là, la ligne suivante (L) est en cours de construction.

[0021] Sur la figure 3, la synchro H correspond à la synchronisation ligne (Horizontale), et la synchro V correspond à la synchronisation trame (Verticale).

[0022] On va maintenant décrire le processus réalisé dans la zone morte verticale VBI pour préparer la construction volatile de la zone active verticale VAI. La première étape peut consister à décoder un descripteur matériel afin d'élaborer des variables qui seront utilisées lors de la construction des lignes pendant la zone VAI.

[0023] Un descripteur matériel est associé à chaque objet. On entend par "descripteur matériel" un ensemble cohérent de données, créé et initialisé généralement par un processus applicatif. Ce descripteur contient toutes les informations pour que l'accélérateur matériel puisse afficher l'objet qui lui est associé. Ces informations, notamment stockées dans des registres ou mémoires, comportent des paramètres graphiques décrivant la nature de l'objet et des paramètres d'affichage. Ces derniers peuvent être dissociés en paramètres indispensables (position, attributs d'affichage tel que niveau de transparence,...) et paramètres de transformations (affichage partiel ou "clipping", re-dimensionnement ou "resizing", rotation, anamorphisme, filtres,...). Chaque objet peut être de nature différente en ce qu'il appartient à une classe donnée (vectoriel, vidéo, bitmap, ...) ou comporte une organisation de couleur donnée (mode palette, noir et blanc, couleurs en 16, 24, 32 bits, avec transparence ou non, ...).

[0024] Le descripteur peut être local à l'accélérateur matériel ou dans une mémoire externe. Dans ce dernier cas, il convient d'abord de récupérer le descripteur avant d'entamer le décodage.

[0025] Le décodage du descripteur consiste à en extraire toutes les variables qui vont servir lors de la construction volatile. Le décodage du descripteur sera plus ou moins long, plus ou moins complexe, en fonction des capacités de l'accélérateur matériel à réaliser des fonctions évoluées et complexes (filtres, "resizing", anamorphisme, "clipping", déplacements,...). Le décodage de certaines informations du descripteur peut même être inutile si les paramètres fournis correspondent déjà aux variables utiles à la construction volatile. Néanmoins, l'idée consiste à déléguer au niveau du matériel, "hardware", le prétraitement des données brutes pour qu'il puisse garantir le temps réel et être synchronisé dans la trame avec la construction volatile. Cela permet aussi de stocker dans le descripteur des paramètres évolués qui seront directement (ou presque) transmis par une application dite API ("Application Program Interface" en langue anglaise) orientée objet. En effet, l'accélérateur matériel est associé à un microprocesseur qui est programmé de manière à offrir un ensemble de fonctions prédéfinies accessibles par l'intermédiaire de cette API.

[0026] Pour illustrer le rôle du décodage, pour un objet composé par exemple d'une image bitmap stockée quelque part en mémoire. Son descripteur contient alors :

- l'adresse où commence la donnée image en mémoire;
- la taille de l'image;
- la position de l'objet à l'écran;
- l'organisation de couleur de l'image bitmap telle qu'elle est stockée en mémoire;
- le niveau de transparence global de l'objet;

[0027] Imaginons que les coordonnées de l'objet soient légèrement négatives par rapport à l'origine de l'écran. Seule la partie visible de l'objet devra être traitée pour être affichée, le reste étant rogné ("clipping"). Cela signifie que la position réelle de l'objet devra être déterminée, que l'adresse de l'objet en mémoire devra être mise à jour pour pointer directement sur le premier pixel visible. La taille du segment de ligne utile à la construction volatile d'une ligne doit aussi être déterminée en fonction du "clipping", mais aussi en fonction de l'organisation de couleur des pixels en mémoire. Ainsi, pour chaque objet, le décodage du descripteur aboutit à une série de variables de travail qui sera exploitée par l'accélérateur matériel lors de la construction volatile. Si les descripteurs peuvent être stockés dans une mémoire externe, les variables utiles seront avantageusement stockées localement à l'accélérateur hardware pour des raisons d'accessibilité. Certaines de ces variables subiront des modifications au fur et à mesure des processus ligne. Par exemple, à la fin du décodage de descripteur, le pointeur d'adresse pointe sur le premier segment de ligne actif de l'objet stocké en mémoire. Mais le pointeur d'adresse devra pointer au fur et à mesure des processus ligne sur les nouveaux segments de ligne actifs.

[0028] Une seconde étape peut être le tri des objets par ordre de profondeur décroissante. Ce processus permet de hiérarchiser l'ordre avec lequel les objets seront superposés à l'écran. Il prend tout son intérêt lorsque la transparence globale des objets est gérée par l'accélérateur matériel puisqu'il permet alors de restituer fidèlement la transparence complexe entre les objets. Cela signifie qu'un objet A placé derrière un objet B sera vu proportionnellement à la transparence de l'objet B qui se trouve devant lui. Le tri pourra se faire, par exemple, en

balayant une première fois tous les objets pour déterminer l'objet le plus enfoui, puis en recommençant le balayage des objets sans tenir compte des objets déjà triés, jusqu'à ce que tous les objets soient triés. A l'issue de ce balayage, chaque objet pourra par exemple contenir dans un de ses registres, l'index de l'objet suivant dans un ordre de profondeur décroissante, afin que le processus ligne de construction de l'image puisse passer simplement d'un objet à un autre, et dans l'ordre exigé par la transparence complexe.

[0029] La transparence complexe n'est obtenue que par le principe inventif de construction volatile selon l'invention et décrit ci-dessous

[0030] On va maintenant décrire le processus réalisé dans la zone active verticale VAI. Le mécanisme de construction volatile d'image est un processus ligne, ce qui veut dire qu'il s'exécute de nouveau pour chaque ligne active de l'écran, c'est-à-dire pour chaque ligne de la VAI (voir figure 3).

Processus ligne :

[0031] Le mécanisme de construction d'une ligne est synchronisé avec le mécanisme d'affichage qui consiste simplement à envoyer à l'écran une suite de pixels à la fréquence pixel. Pour ce faire, on peut très bien n'utiliser qu'une seule mémoire de ligne pour ces deux mécanismes, sachant que le mécanisme de construction doit alors être temporisé pour ne pas mettre à jour des pixels qui n'ont pas encore été envoyés à l'écran. On peut également utiliser deux ou plusieurs mémoires de ligne. Dans ce cas, certaines dites "off screen" servant à construire les lignes suivantes, pendant que une dite "on screen" est envoyée à l'écran pour remplir la ligne courante. A la fréquence ligne, par exemple au moment du top de synchro ligne, les processus sont inversés. La mémoire "off screen" devient la mémoire "on screen" et vice et versa. La figure 4 illustre ce mécanisme sur deux lignes successives Ligne n et Ligne n+1 où l'on remarque l'inversion des rôles entre les deux mémoires 10 et 11.

Construction de la ligne :

[0032] Pour une ligne de l'écran donnée, il s'agit de récupérer tous les segments des objets qui doivent se trouver dans cette ligne.

[0033] Pour savoir si un objet se trouve dans la ligne, on exécute un processus permettant successivement pour chaque objet d'identifier si oui ou non l'objet est présent dans la ligne d'écran considéré. Le fait qu'un objet soit présent ou non dans une ligne de l'écran dépend de plusieurs paramètres qui sont propres à chaque objet. Dans le cas d'un objet bitmap rectangulaire stocké comme tel et affiché comme tel, on peut citer :

- la hauteur de l'objet (en nombre de pixels);
- la coordonnée "y" de l'objet (en nombre de pixels par rapport à une origine conventionnelle).

[0034] Si l'objet est affecté d'une zone de "clipping", on tiendra compte également des paramètres de cette zone.

[0035] Si l'objet doit subir un "resizing" vertical, on tiendra compte de la taille finale de l'objet, c'est-à-dire à l'écran, pour déterminer si un morceau de cet objet se trouve dans la ligne d'écran considérée.

[0036] Lorsque l'on sait qu'un objet est présent dans la ligne active, il faut encore déterminer quelle est la zone utile de cet objet dans la ligne considérée, savoir où et comment récupérer cette zone, et savoir à quel endroit la placer dans la mémoire de ligne. Par exemple, dans le cas d'un objet bitmap rectangulaire stocké en mémoire et affiché anamorphosé à l'écran, la récupération du segment utile oblige entre autre à déterminer :

- l'adresse mémoire où se trouve le premier pixel du segment d'objet;
- le nombre de pixels contenu dans le segment de l'objet;
- la loi de variation horizontale et verticale entre deux pixels.

[0037] La détermination de ces variables ne suffit pas à récupérer le segment utile de l'objet. Il faut également considérer les paramètres objet suivants :

- la largeur de l'objet stocké en mémoire (en octets);
- l'organisation de couleur des pixels;
- la loi de compression, si l'objet est stocké de façon compressée.

[0038] Du point de vue des actions à effectuer : on exécute un processus permettant successivement pour chaque objet :

a) de calculer les variables nécessaires à la récupération du segment utile;
b) de récupérer les paramètres objet nécessaires;
c) de mettre à jour les variables dans les registres en anticipation des prochaines lignes, par exemple le pointeur d'adresse mémoire.

[0039] On exécute également un processus permettant d'accéder à la mémoire où sont stockés les objets, à partir de commandes matérielles. On peut par exemple parler de "DMA hardware", pour "Direct Memory Access" en langue anglaise, géré par l'accélérateur matériel, par comparaison avec un DMA traditionnellement géré par un microprocesseur.

[0040] On exécute aussi un processus permettant de commander et de surveiller le DMA.

[0041] Le mécanisme "hardware" de construction volatile selon l'invention peut avantageusement être implémenté dans un FPGA qui permet d'intégrer la totalité des modules et processus nécessaires à la mise en oeuvre du procédé selon l'invention. Sur la figure 5, on voit la manière dont un accélérateur matériel 21 peut être ar-

chitecturé sur un tel FPGA.

[0042] Pour le stockage des objets graphiques, on utilise avantageusement une mémoire 20 (SDRAM, DDRAM...) dont la bande-passante est suffisamment importante pour permettre au mécanisme de construction volatile de récupérer suffisamment d'informations (données utiles des objets actifs) lors du processus ligne. Cette mémoire 20 est une mémoire vive externe à l'accélérateur 21.

[0043] Sur la figure 5, on distingue :

a) Un gestionnaire d'objet 14 réalisant l'activation et la caractérisation des traitements propres à l'objet t (intra-objet) ainsi que la gestion entre les objets (inter-objets).

b) Un registre 15 de stockage des paramètres et variables de travail pour chaque objet.

c) Un module "DMA hardware" 12 contrôlé par le gestionnaire d'objet 14 et capable d'aller rechercher les données (par exemple des images bitmap) dans la mémoire externe 20.

d) Une mémoire tampon 13 pour stocker temporairement les données brutes en provenance du DMA 12, lorsque les processus DMA et un pipeline 16 de décompression/conversion ne sont pas synchronisés.

e) Deux mémoires de ligne 19 ou "line buffers" en langue anglaise, (l'une "off screen" et l'autre "on screen" conformément à la figure 4.

f) Le pipeline 16 de décompression/conversion des données brutes en pixels. Plus précisément, Une fois que le segment d'objet est récupéré par le DMA, il s'agit de convertir les données brutes issues du DMA en pixels dans le format de sortie retenu pour pouvoir être stockés dans la mémoire de ligne. Dans le cas d'un objet de type image bitmap, cela nécessite de connaître :

- l'organisation de couleur des pixels;
- la loi de variation horizontale entre deux pixels; et
- la loi de compression, si l'image est stockée de façon compressée.

Ce pipeline comprend également des moyens pour transformer les pixels. Une fois les données brutes converties en pixels, il est possible d'appliquer des traitements numériques, tels que par exemple des filtres ou des effets sur ces pixels. On peut par exemple citer :

- "chroma-keying" : le pixel devient transparent si sa valeur est égale à une valeur de référence qu'on appelle la valeur de "chroma-key";
- seuillage par luminance : le pixel devient transparent si la valeur de son niveau de luminance est inférieure à une valeur de référence qu'on appelle valeur de seuil;
- seuillage par chrominance;
- niveau de transparence : on modifie le niveau de transparence du pixel dit "alpha" en appliquant la formule alpha_pixel = alpha_global_objet x alpha_pixel_source;
- filtres de couleur : modification de la chrominance du pixel;
- filtres passe-bas, passe-haut : influence des pixels voisins au pixel courant par un pipeline adapté;
- Etc.

g) Un mélangeur de pixel 18 ("blending")suivant la transparence du pixel considéré. L'écriture des pixels en mémoire ligne représente la dernière étape dans le processus ligne de la construction volatile. L'écriture d'un pixel dans la mémoire nécessite de connaître :

- la position du pixel dans la ligne : coordonnée "x" par rapport à l'origine égale au premier pixel de l'écran; et
- la valeur du pixel déjà présent dans la ligne à la même position, en particulier lorsque l'objet est semi-transparent.

h) Un multiplexeur 17 sélectionne soit le pixel issu du flux vidéo principal (vidéo_in), soit le pixel issu du flux provenant du DMA 12. Ce mécanisme impose que le cadencement de l'accélérateur soit synchronisé et verrouillé sur la source vidéo. La fréquence de fonctionnement du pipeline est égale à plusieurs fois la fréquence pixel de l'écran, et le multiplexeur sélectionne le pixel vidéo à la cadence d'une fois par période pixel (à la fréquence pixel). Le reste du temps, le multiplexeur gère le flux issu du DMA 12. Lorsqu'il n'y a pas de flux vidéo principal, celui-ci est remplacé par une couleur de fond ("background"). Cet exemple d'architecture permet aussi de réaliser une fusion de la vidéo avec la couleur de fond proportionnellement à un niveau de transparence vidéo (alpha_vidéo_in) sans implémenter un deuxième module de fusion.

[0044] Ainsi, le procédé de construction d'image selon l'invention rend possible la gestion de niveau de profondeur entre objets graphiques et vidéo sans limite du nombre maximal de couches. Ce nombre de couches n'est pas dimensionnant au niveau des ressources matérielles de l'accélérateur. Ce procédé rend également possible une gestion de la transparence entre les objets graphiques et vidéo en fonction du positionnement des objets sur l'axe des z quel que soit le nombre de couches graphiques qui n'est pas dimensionnant pour obtenir la transparence complexe globale.

Revendications

1. Procédé pour construire une image apte à être affichée sur un système d'affichage à partir d'une pluralité d'objets et de descripteurs, chaque descripteur étant associé à chaque objet, un descripteur contenant des paramètres graphiques décrivant la nature de l'objet associé et des paramètres d'affichage pour afficher l'objet associé, pour chaque trame vidéo du système d'affichage, l'image est construite ligne par ligne en réalisant les étapes suivantes :

- pour chaque ligne du système d'affichage, construction à la volée de cette ligne en récupérant et en stockant, en temps réel, dans une mémoire ligne, tous pixels relatifs aux objets destinés à être affichés sur ladite ligne, pour ce faire le procédé comprend en outre les étapes suivantes :

  - identification de façon indépendante de chaque objet devant être présent sur cette ligne en fonction d'un ensemble de variables propres à chaque objet ;
  - détection dans chaque objet ainsi identifié, d'une zone utile correspondant à la ligne considérée ; et
  - conversion des données brutes issues de ladite zone utile en pixels compatibles avec le format d'affichage,
  - pour chaque pixel de la zone utile et pour chaque objet identifié, écriture d'un pixel dans la mémoire ligne, ce pixel étant destiné à s'afficher sur une position donnée sur une ligne du système d'affichage, la position de ce pixel est d'abord déterminée dans la ligne considérée, et ce premier pixel est mélangé avec un second pixel actuellement stocké dans la mémoire ligne à ladite même position donnée, le mélangeur étant appliqué de manière proportionnelle au niveau de transparence dudit premier pixel;

- envoi de ces pixels au système d'affichage selon une suite de pixels, de sorte que l'image complète n'est formée que sur ledit système d'affichage ; le mécanisme de construction de la ligne étant synchronisé en ligne et en trame avec le système d'affichage.

2. Procédé selon la revendication 1, **caractérisé en ce qu'**il comprend l'étape d'utilisation de deux mémoires lignes réalisant, de façon décalée, chacune successivement une étape de construction puis une étape d'affichage de telle sorte que lorsqu'une première mémoire ligne affiche des pixels sur la ligne courante, on construit et stocke dans la seconde mémoire ligne des pixels destinés à être affichés sur la ligne suivante.

3. Procédé selon la revendication 1 ou 2, **caractérisé en ce qu'**au cours de l'étape de détection, accès à la mémoire où sont stockés les objets à partir de mécanismes électroniques et stockage temporaire des données brutes issues des zones utiles dans des moyens de stockage.

4. Procédé selon l'une quelconque des revendications précédentes, caractérisé en qu'une fois les données brutes converties en pixels, application des transformations et des effets à ces pixels.

5. Procédé selon l'une quelconque des revendications précédentes, **caractérisé en ce que** la trame vidéo comprenant deux zones temporelles distinctes que sont une zone morte verticale, dite "verticale blanking intervalle VBI" en langue anglaise et une zone active verticale, dite "verticale active intervalle VAI" en langue anglaise, correspondant respectivement à l'intervalle entre les deux trames actives et à la durée d'affichage d'une trame active ; on réalise au cours de chaque zone active verticale, toute préparation nécessaire au déroulement de la construction, les étapes de construction et d'affichage à la volée.

6. Procédé selon la revendication 5, **caractérisé en ce que** ladite préparation comprend en outre un tri des objets par ordre de profondeur de façon à hiérarchiser l'ordre avec lequel lesdits objets seront superposés sur le système d'affichage.

7. Procédé selon la revendication 6, **caractérisé en ce que** l'accélérateur matériel est apte à générer la trame vidéo du système d'affichage.

8. Procédé selon la revendication 7, **caractérisé en ce que** l'accélérateur matériel est apte à se synchroniser et se verrouiller sur la trame vidéo du système d'affichage à partir d'une information de synchronisation.

9. Procédé selon l'une quelconque des revendications précédentes, **caractérisé en ce que** le système d'affichage comprend plusieurs écrans synchronisés en ligne et en trame, et pilotés par un contrôleur unique générant des signaux de cadencement des écrans, ce contrôleur étant disposé dans un accélérateur.

10. Système pour construire une image apte à être affichée sur un système d'affichage à partir d'une pluralité d'objets et de descripteurs, et en ce que chaque descripteur est associé à chaque objet, un descripteur contenant des paramètres graphiques décrivant la nature de l'objet associé et des paramètres d'affichage pour afficher l'objet associé, ce système comprenant une unité de traitement associé à un

accélérateur matériel ; l'accélérateur matériel étant utilisé lors de la construction pour construire l'image ligne par ligne directement depuis les objets au moyen de mécanismes électroniques par des machines d'états, l'accélérateur matériel étant en outre configuré pour réaliser les étapes telles que définies dans les revendications précédentes.

11. Système selon la revendication 10, **caractérisé en ce que** l'accélérateur matériel comprend les éléments suivants :

- un gestionnaire d'objets pour activer et caractériser des traitements liés aux objets,
- une mémoire de stockage des paramètres et variables de chaque objet,
- un module matériel de type DMA ("Direct Memory Access" en langue anglaise) contrôlé par l'accélérateur et apte à récupérer des données des objets depuis une mémoire,
- une mémoire tampon pour stocker temporairement les données provenant du module matériel DMA,
- un module de décompression et de transformation de données brutes provenant de la mémoire tampon en pixels,
- un multiplexeur pour sélectionner le pixel à afficher entre celui provenant du module de décompression et de transformation et celui provenant d'une source externe,
- un mélangeur pour mélanger le pixel provenant du multiplexeur et un pixel actuellement affiché en fonction de la transparence du pixel provenant du multiplexeur, et
- deux mémoires ligne assurant successivement à tour de rôle l'affichage sur la ligne courante du système d'affichage de pixels préalablement stockés, et le stockage de pixels provenant du mélangeur et qui seront affichés à la ligne suivante.

12. Système selon la revendication 10 ou 11, **caractérisé en ce qu'**il comprend des moyens pour se synchroniser sur une source vidéo et insérer des informations graphiques dans un flux vidéo de ladite source vidéo, non stocké en mémoire.

Patentansprüche

1. Verfahren zur Erstellung eines auf einem Anzeigesystem anzeigbaren Bildes aus einer Vielzahl von Objekten und Deskriptoren, wobei jeder Deskriptor jedem Objekt zugeordnet ist, wobei ein Deskriptor Grafikparameter, die die Art des zugeordneten Objektes beschreiben, und Anzeigeparameter zum Anzeigen des zugeordneten Objekts enthält, für jedes Video-Frame des Anzeigesystems wird das Bild Zeile für Zeile unter Durchführen der folgenden Schritte erstellt:

- für jede Zeile des Anzeigesystems, Erstellen dieser Zeile on-the-fly, indem alle Pixel bezüglich der Objekte, die dazu bestimmt sind, in der Zeile angezeigt zu werden, gesammelt und in Echtzeit in einem Zeilenspeicher gespeichert werden, hierfür umfasst das Verfahren ferner die folgenden Schritte:

  - unabhängiges Identifizieren eines jeden Objektes, das in dieser Zeile vorhanden sein soll, in Abhängigkeit von einer Menge von einem jedem Objekt eigenen Variablen;
  - Erfassen eines der betrachteten Zeile entsprechenden nützlichen Bereichs in jedem auf diese Weise identifizierten Objekt; und
  - Umwandeln der aus dem nützlichen Bereich stammenden Rohdaten in mit dem Anzeigeformat kompatible Pixel,
  - für jedes Pixel des nützlichen Bereichs und für jedes identifizierte Objekt, Schreiben eines Pixels in den Zeilenspeicher, wobei dieses Pixel dazu bestimmt ist, in einer gegebenen Position in einer Zeile des Anzeigesystems angezeigt zu werden, die Position dieses Pixels wird zunächst in der betrachteten Zeile bestimmt, und dieses erste Pixel wird mit einem aktuell in dem Zeilenspeicher in der gleichen gegebenen Position gespeicherten zweiten Pixel gemischt, wobei der Mischer proportional auf den Transparenzwert des ersten Pixels angewandt wird;

- Senden dieser Pixel an das Anzeigesystem entsprechend einer Pixelfolge, so dass das vollständige Bild nur auf dem Anzeigesystem gebildet wird; wobei der Zeilenerstellmechanismus mit dem Anzeigesystem zeilen- und framesynchronisiert ist.

2. Verfahren nach Anspruch 1, **dadurch gekennzeichnet, dass** es den Schritt der Verwendung von zwei Zeilenspeichern umfasst, die versetzt jeweils nacheinander einen Erstellungsschritt, dann einen Anzeigeschritt derart durchführen, dass wenn ein erster Zeilenspeicher Pixel auf der akutellen Zeile anzeigt, in dem zweiten Zeilenspeicher Pixel erstellt und gespeichert werden, die dazu bestimmt sind, in der folgenden Zeile angezeigt zu werden.

3. Verfahren nach Anspruch 1 oder 2, **dadurch gekennzeichnet, dass** im Laufe des Erfassungsschrittes anhand von elektronischen Mechanismen auf den Speicher, in dem die Objekte gespeichert sind, zugegriffen wird und die aus den nützlichen Bereichen stammenden Rohdaten in Speichermitteln

temporär gespeichert werden.

4. Verfahren nach einem der vorhergehenden Ansprüche, **dadurch gekennzeichnet, dass** sobald die Rohdaten in Pixel umgewandelt sind, Transformationen und Effekte auf diese Pixel angewandt werden.

5. Verfahren nach einem der vorhergehenden Ansprüche, **dadurch gekennzeichnet, dass**, da das Video-Frame zwei getrennte Zeitbereiche umfasst, dies sind ein toter vertikaler Bereich, im Englischen als "vertical blanking interval VBI" bezeichnet, und ein aktiver vertikaler Bereich, im Englischen als "vertical active interval VAI" bezeichnet, die dem Abstand zwischen den beiden aktiven Frames bzw. der Anzeigedauer eines aktiven Frames entsprechen, im Laufe eines jeden aktiven vertikalen Bereichs jede für das Ablaufen der Erstellung erforderliche Vorbereitung, die Schritte des Erstellens und Anzeigens on-the-fly vollzogen werden.

6. Verfahren nach Anspruch 5, **dadurch gekennzeichnet, dass** die Vorbereitung ferner ein Sortieren der Objekte nach Tiefe umfasst, um die Reihenfolge zu hierarchisieren, mit der die Objekte auf dem Anzeigesystem übereinander angeordnet werden.

7. Verfahren nach Anspruch 6, **dadurch gekennzeichnet, dass** der Hardwarebeschleuniger geeignet ist, das Video-Frame des Anzeigesystems zu generieren.

8. Verfahren nach Anspruch 7, **dadurch gekennzeichnet, dass** der Hardwarebeschleuniger geeignet ist, anhand einer Synchronisierungsinformation auf das Video-Frame des Anzeigesystems synchronisiert und hieran verriegelt zu werden.

9. Verfahren nach einem der vorhergehenden Ansprüche, **dadurch gekennzeichnet, dass** das Anzeigesystem mehrere Bildschirme umfasst, die zeilenund framesynchronisiert und durch einen einzigen Controller, welcher Signale zur Taktung der Bildschirme erzeugt, gesteuert sind, wobei dieser Controller in einem Beschleuniger angeordnet ist.

10. System zur Erstellung eines auf einem Anzeigesystem anzeigbaren Bildes aus einer Vielzahl von Objekten und Deskriptoren, und dass jeder Deskriptor jedem Objekt zugeordnet ist, wobei ein Deskriptor Grafikparameter, die die Art des zugeordneten Objektes beschreiben, und Anzeigeparameter zum Anzeigen des zugeordneten Objekts enthält, wobei dieses System eine Verarbeitungseinheit, welche einem Hardwarebeschleuniger zugeordnet ist, umfasst; wobei der Hardwarebeschleuniger bei der Erstellung verwendet wird, um das Bild Zeile für Zeile direkt aus den Objekten mittels elektronischer Mechanismen durch Zustandsautomaten zu erstellen, wobei der Hardwarebeschleuniger ferner dazu ausgelegt ist, die Schritte, wie sie in den vorhergehenden Ansprüchen definiert sind, durchzuführen.

11. System nach Anspruch 10, **dadurch gekennzeichnet, dass** der Hardwarebeschleuniger die folgenden Elemente umfasst:

- einen Objekt-Manager, um mit den Objekten verbundene Verarbeitungen zu aktivieren und zu kennzeichnen,
- einen Speicher zum Speichern der Parameter und Variablen eines jeden Objekts,
- ein Hardwaremodul vom Typ DMA ("Direct Memory Access" im Englischen), das durch den Beschleuniger gesteuert wird und geeignet ist, Daten der Objekte aus einem Speicher zu sammeln,
- einen Pufferspeicher, um die aus dem DMA-Hardwaremodul stammenden Daten temporär zu speichern,
- ein Modul zum Dekomprimieren und zum Umwandeln von aus dem Pufferspeicher stammenden Rohdaten in Pixel,
- einen Multiplexer zum Auswählen des anzuzeigenden Pixels zwischen dem aus dem Dekompressions- und Umwandlungsmodul stammenden und dem aus einer externen Quelle stammenden,
- einen Mischer zum Mischen des aus dem Multiplexer stammenden Pixels und eines aktuell angezeigten Pixels in Abhängigkeit von der Transparenz des aus dem Multiplexer stammenden Pixels, und
- zwei Zeilenspeicher, die nacheinander abwechselnd die Anzeige von zuvor gespeicherten Pixeln in der aktuellen Zeile des Anzeigesystems und die Speicherung von aus dem Mischer stammenden Pixeln, welche in der folgenden Zeile angezeigt werden, sicherstellen.

12. System nach Anspruch 10 oder 11, **dadurch gekennzeichnet, dass** es Mittel umfasst, um auf eine Videoquelle synchronisiert zu werden und um Grafikinformationen in einen Videostream der Videoquelle, der nicht im Speicher abgelegt ist, einzufügen.

Claims

1. Method of constructing an image suitable for being displayed on a display system from a plurality of objects and descriptors, each descriptor being associated with each object, a descriptor including graphic parameters defining the nature of the associated object and display parameters for displaying the asso-

ciated object, for each video frame of the display system, the image is constructed line-by-line by carrying out the following steps:

- for each line of the display system, on-the-fly construction of this line by retrieving and storing, in real time, in one line buffer, all the pixels relating to the objects intended to be displayed on said line, in order to do this the method further comprising following step:

  - identification indenpendently of each object that must be present on this line according to a set of variables specific to each object;
  - detection in each object thus identified, a useful zone corresponding to the considered line; and
  - conversion of the raw data coming from said useful zone into pixels compatible with the display format,
  - for each pixel of the useful zone and for each identified object, writing a pixel in the line buffer, this pixel is intended to be displayed on a given position on a line of the display system, firstly the position of this pixel in the considered line is determined, and this first pixel is blended with a second pixel currently stored in the line buffer at said same given position, the blender being applied proportionally to the level of transparency of said first pixel;
  - sending these pixels to the display system according to a pixels sequence, such that the complete image is formed only on said display system; the line construction mechanism being line- and frame-synchronized with the display system.

2. Method according to claim 1, **characterized in that** it comprises the step of using two line buffers, each successively carrying out, in a shifted manner, a construction step then a display step such that, when a first line buffer displays pixels on the current line, pixels intended to be displayed on the following line are constructed and stored in the second line buffer.

3. Method according to claim 1 or 2, **characterized in that** during the detection step, accessing the memory where the objects are stored from electronic mechanisms and the raw data from the useful intervals is stored temporarily in storage means.

4. Method according to any one of the preceding claims, **characterized in that** once the raw data is converted into pixels, application of transformations and effects to these pixels.

5. Method according to any one of preceding claims, **characterized in that** the video frame comprises two separate time intervals, a so-called "vertical blanking interval" (VBI) and a so-called "vertical active interval" (VAI), respectively corresponding to the interval between the two active frames and to the period of display of an active frame, any preparation necessary to the progress of the construction, the construction and display steps are carried out instantly during each vertical active interval.

6. Method according to claim 5, **characterized in that** said preparation also comprises a sorting of the objects in order of depth so as to hierarchize the order with which the objects will be overlaid on the display system.

7. Method according to claim 6, **characterized in that** hardware accelerator is suitable for generating the video frame of the display system.

8. Method according to claim 7, **characterized in that** the hardware accelerator is suitable for synchronizing and locking on the video frame of the display system from synchronization information.

9. Method according to any one of the preceding claims, **characterized in that** the display system comprises several line- and frame-synchronized screens, and driven by a single controller generating signals that clock from the screens, this controller being located in an accelerator.

10. System for constructing an image suitable to be displayed on a display system from a plurality of objects and descriptors, and in that each descriptor is associated with each object, a descriptor including graphic parameters defining the nature of the associated object and display parameters for displaying the associated object, this system comprising a processing unit combined with a hardware accelerator; the hardware accelerator being used during the construction for constructing the image line by line directly from the objects by means of electronic mechanisms by sate machines, the hardware accelerator being also configured to carry out steps as defined in preceding claims.

11. System according to claim 10, **characterized in that** the hardware accelerator comprises the following elements:

- an object manager for activating and characterizing operations linked to the objects,
- a memory for storing the parameters and variables of each object,
- a DMA (Direct Memory Access)-type hardware module controlled by the accelerator and suita-

ble for retrieving data about the objects from a memory,
- a buffer for temporarily storing the data from the DMA hardware module,
- a module decompressing and converting raw data from the buffer as pixels,
- a multiplexer for selecting the pixel to be displayed between that from the decompression and conversion module and that from an external source,
- a blender for blending the pixel from the multiplexer and a currently displayed pixel according to the transparency of the pixel from the multiplexer, and
- two line buffers successively carrying out in turn the display of previously stored pixels on the current line of the display system, and the storage of pixels from the blender which will be displayed on the following line.

12. System according to claim 10 or 11, **characterized in that** it comprises means for synchronizing on a video source and inserting graphic information into a video flow of said video source, not stored in the memory.

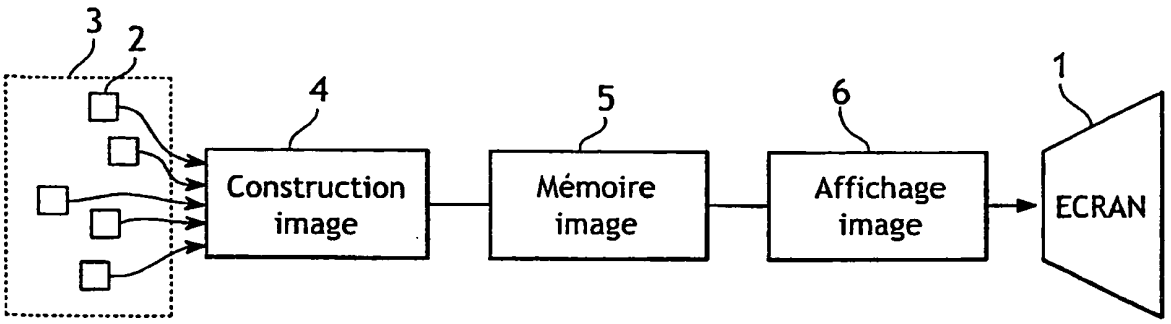

FIG.1

ART ANTERIEUR

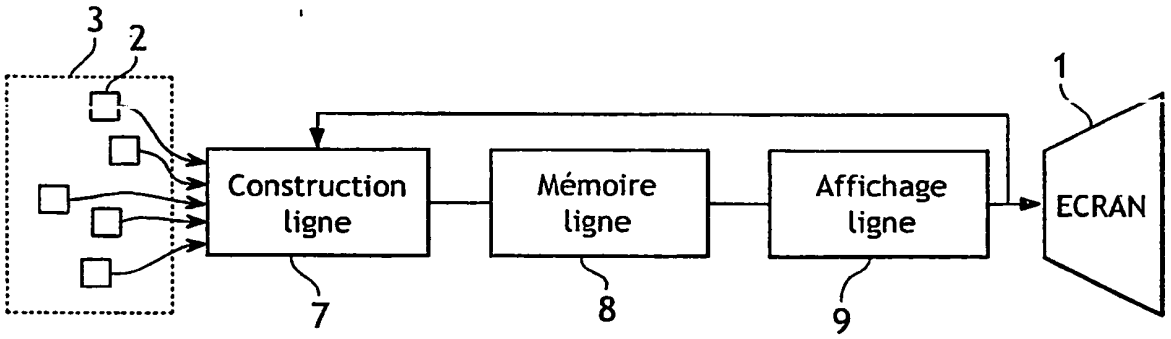

FIG.2

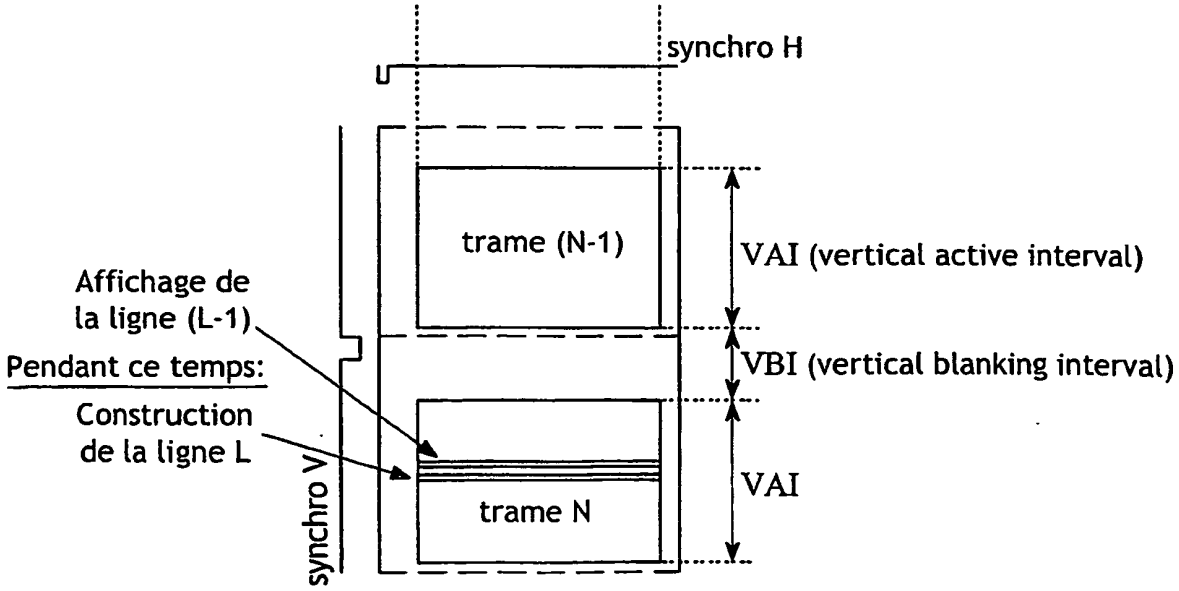

FIG.3

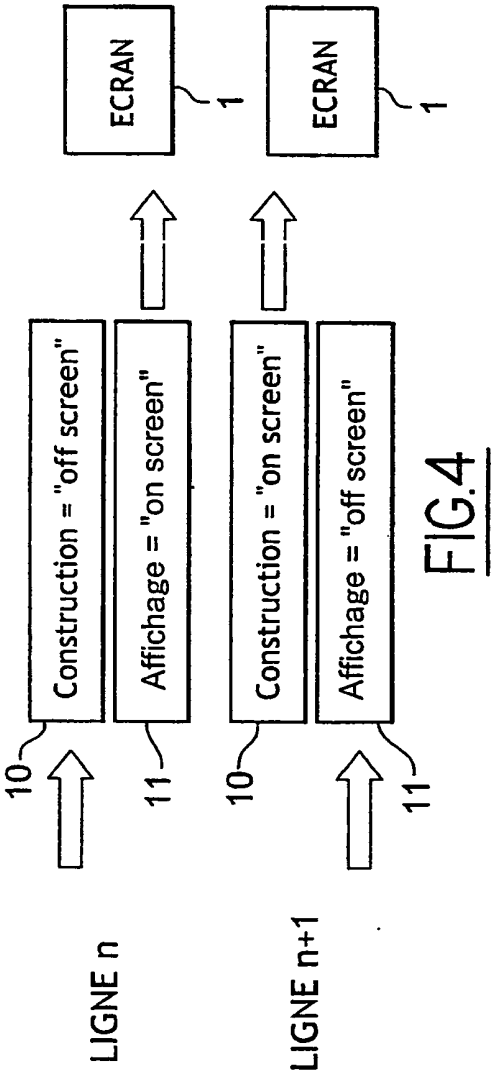
LIGNE n
LIGNE n+1
Construction = "off screen"
Affichage = "on screen"
Construction = "on screen"
Affichage = "off screen"
ECRAN
ECRAN
1
1
10
11
10
11

FIG.4

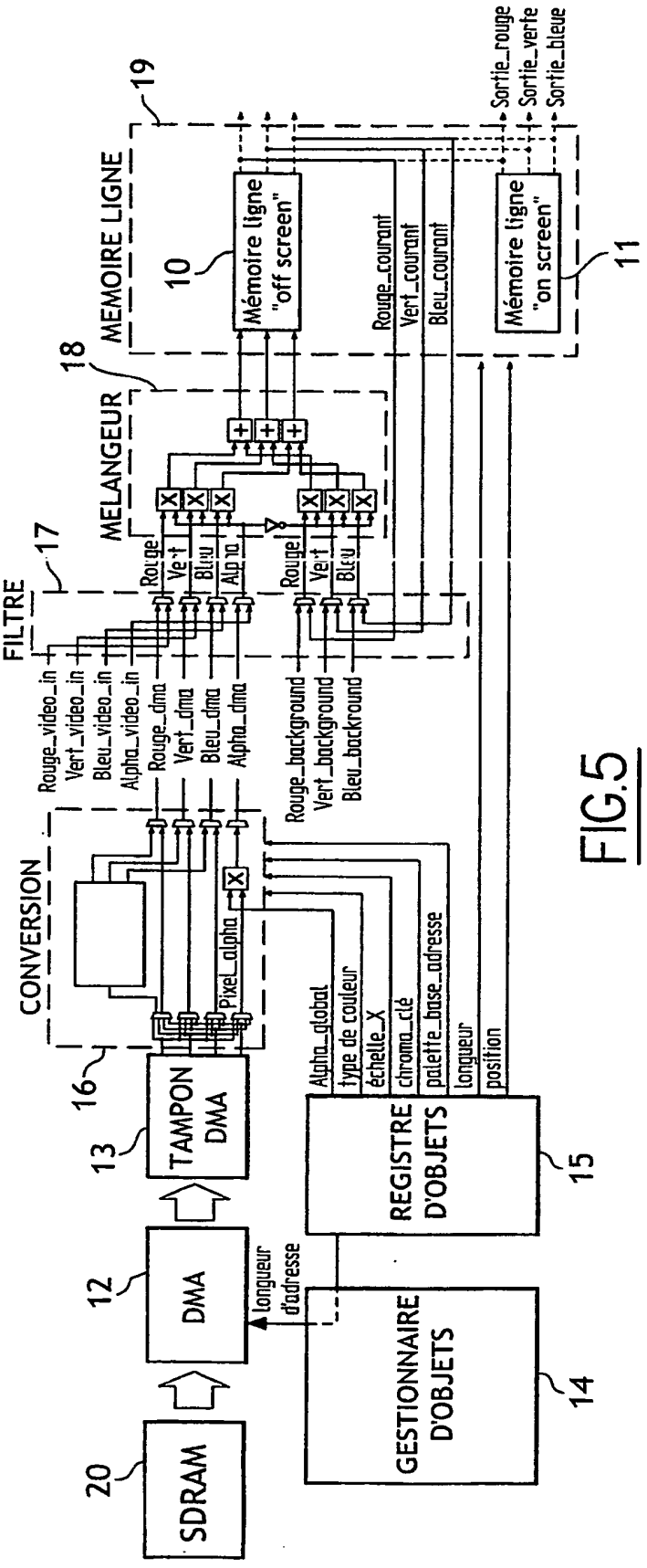
MEMOIRE LIGNE
19
Mémoire ligne "off screen"
Mémoire ligne "on screen"
10
11
MELANGEUR
18
FILTRE
17
CONVERSION
16
TAMPON DMA
13
DMA
12
SDRAM
20
REGISTRE D'OBJETS
15
GESTIONNAIRE D'OBJETS
14
Rouge_courant
Vert_courant
Bleu_courant
Sortie_rouge
Sortie_verte
Sortie_bleue
Rouge_video_in
Vert_video_in
Bleu_video_in
Alpha_video_in
Rouge_dma
Vert_dma
Bleu_dma
Alpha_dma
Pixel_alpha
Rouge_background
Vert_background
Bleu_background
Alpha_global
type de couleur
échelle_X
chroma_clé
palette_base_adresse
longueur
position
Longueur d'adresse

FIG.5