(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
18 octobre 2012 (18.10.2012)

WIPO | PCT

(10) Numéro de publication internationale
WO 2012/140361 A1

(51) Classification internationale des brevets : *G06T 15/00* (2011.01)

(21) Numéro de la demande internationale : PCT/FR2012/050784

(22) Date de dépôt international : 11 avril 2012 (11.04.2012)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
11/53173 12 avril 2011 (12.04.2011) FR

(71) Déposant *(pour tous les États désignés sauf US)* : **REAL FUSIO FRANCE** [FR/FR]; 10 Rue des Arts, F-31000 Toulouse (FR).

(72) Inventeurs; et

(75) Inventeurs/Déposants *(pour US seulement)* : **MACHADO, Abilio** [FR/FR]; 91 rue Fondeville, F-31000 Toulouse (FR). **GERMAIN, Marc** [FR/FR]; 52 avenue Jean Chaubet, F-31500 Toulouse (FR).

(74) Mandataire : **Cabinet GERMAIN & MAUREAU**; B.P.6153, F-69466 LYON Cedex 06 (FR).

(81) États désignés *(sauf indication contraire, pour tout titre de protection nationale disponible)* : AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) États désignés *(sauf indication contraire, pour tout titre de protection régionale disponible)* : ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), européen (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Publiée :

— *avec rapport de recherche internationale (Art. 21(3))*

(54) Title : METHOD AND SYSTEM FOR RENDERING A VIRTUAL SCENE IN THREE DIMENSIONS

(54) Titre : PROCÉDÉ ET SYSTÈME DE RENDU D'UNE SCÈNE VIRTUELLE EN TROIS DIMENSIONS

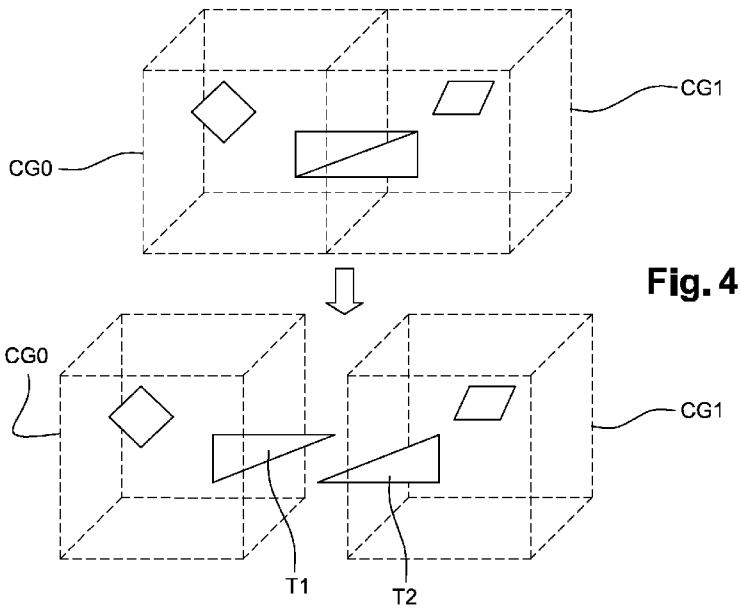


Fig. 4

(57) Abrégé :

(57) Abstract : The invention relates to a method for rendering a virtual scene in three dimensions, comprising: a first data preparation phase including at least one step in which the data of the scene are grouped into a data structure (OD) corresponding to a spatial division of the scene into a regular grid, such as to form metavoxels corresponding to the separation of a scene model into a set of submodels according to the splitting of said regular grid; and a second real-time rendering phase using the geometric data structure (OD) in order to render at least one image. The above-mentioned first data preparation phase includes at least one step comprising the injection of keys relating to metadata values into the data structure (OD) and the compiling of a database containing the values of the metadata corresponding to the keys stored in the geometric data structure (OD), such that, during a user interaction in the second rendering phase, a metadatum value can be made available that has been extracted from the base linked with a relevant image element corresponding to the rendering of a metavoxel.

[Suite sur la page suivante]

WO 2012/140361 A1

La présente invention concerne un procédé de rendu d'une scène virtuelle en trois dimensions comprenant: une première phase de préparation des données comprenant au moins une étape de regroupement des données de la scène dans une structure de données (OD) correspondant à une division spatiale de la scène en une grille régulière de façon à former des métavoxels correspondant à la séparation d'un modèle de scène en un ensemble de sous- modèles suivant le découpage de ladite grille régulière, et une seconde phase de rendu temps-réel utilisant la structure de donnée géométrique (OD) pour le rendu d'au moins une image, la première phase de préparation des données comprenant au moins une étape d'injection de clés relatives à des valeurs de métadonnées dans ladite structure de donnée (OD) et la constitution d'une base de donnée comprenant les valeurs des métadonnées correspondants aux clés stockées dans la structure de donnée géométriques (OD), de façon à permettre, lors d'une interaction utilisateur dans la seconde phase de rendu, la mise à disposition d'une valeur de métadonnée extraite de la base en lien avec un élément d'image considéré correspondant au rendu d'un métavoxel.

Procédé et système de rendu d'une scène virtuelle en trois dimensions

Domaine technique et état de la technique

La présente invention a pour objet un procédé de rendu utilisant un chargement partiel en mémoire vive pour des modèles en trois dimensions de très grande taille.

Les besoins d'affichage de données numériques massives concernent des applications dans lesquelles des données numériques de très grande taille sont utilisées, par exemple dans le cas de maquette numérique utilisées dans la CAO ou l'architecture, dans le cas de systèmes d'information géographiques représentant des terrains ou une cartographie, dans le cas d'application du domaine de la conservation du patrimoine pour des modèles de monuments ou d'œuvres d'art, dans le domaine de l'imagerie médicale et notamment dans l'utilisation de scanners.

Il convient de préserver autant que possible la fidélité de l'affichage, en opérant le cas échéant une dégradation minimisée tout en maintenant une performance de rafraîchissement acceptable pour une application temps réel (rendu à une fréquence supérieure à 10 images par secondes).

A titre d'exemple, il est connu du document US6933946 de procéder à un rendu de type utilisant un chargement partiel en mémoire vive avec une phase de préparation des données.

Ce type de procédé donne satisfaction en ce qu'il permet effectivement de réaliser un rendu d'ensemble de données 3D de très grande taille.

Il apparaît toutefois que ce type de procédé n'est pas adapté à un parc de machines hétéroclites aux capacités limitées, et comprenant notamment une taille de la mémoire vive (RAM) inférieure à 3GB sur des systèmes 32bits, présentant ou non un matériel d'accélération de l'affichage 3D de type carte graphique (GPU).

Dans le cas ou un tel matériel est présent, sa capacité maximale effective d'affichage est limitée, de l'ordre de 30 millions de triangles par

seconde à l'heure actuelle, et la mémoire vidéo (VRAM) est également limitée, par exemple à une valeur de l'ordre de 2 Go.

Bien entendu, les progrès matériels permettent d'envisager des augmentations de ces capacités, toutefois ces progrès ne concernent pas le parc installé et ne permettent pas d'envisager à moyen terme une prise en charge de l'affichage d'ensemble massif de données 3D sur un matériel informatique moyen et d'entrée de gamme.

Exposé de l'invention

La présente invention a pour but de résoudre tout ou partie des inconvénients mentionnés ci-dessus.

A cet effet, la présente invention concerne un procédé de rendu d'une scène virtuelle en trois dimensions comprenant:

une première phase de préparation des données comprenant au moins une étape de regroupement des données de la scène dans une structure de données correspondant à une division spatiale de la scène en une grille régulière de façon à former des métavoxels correspondant à la séparation d'un modèle de scène en un ensemble de sous-modèles suivant le découpage de ladite grille régulière , et

une seconde phase de rendu temps-réel utilisant la structure de donnée géométrique pour le rendu d'au moins une image,

la première phase de préparation des données comprenant au moins une étape d'injection de clés relatives à des valeurs de métadonnées dans ladite structure de donnée et la constitution d'une base de donnée comprenant les valeurs des métadonnées correspondants aux clés stockées dans la structure de donnée géométriques, de façon à permettre, lors d'une interaction utilisateur dans la seconde phase de rendu, la mise à disposition d'une valeur de métadonnée extraite de la base en lien avec un élément d'image considéré correspondant au rendu d'un métavoxel.

Grâce aux dispositions selon l'invention, un rendu fidèle de données 3D numérique massive est obtenue. Le système permet de prendre en compte tout type de donnée numérique et de géométrie (maillage triangulaire, surface paramétrique, voxels, brep...) ainsi que l'insertion de métadonnées, c'est-à-dire de données autres que la géométrie, comme des propriétés, des données sur les matériaux, le producteur et la date de génération d'un élément de la géométrie.

Les dispositions selon l'invention permettent également de limiter les pertes lors de la conversion et lors de l'affichage. Les dispositions selon l'invention n'imposent pas de contrainte sur la taille de l'ensemble de données 3D, permettent un chargement dynamique de cet ensemble de données 3D, et un fonctionnement hors cœur (« out-of-core »), les données étant partiellement chargées en mémoire vive (RAM).

L'invention concerne également un système destiné à la mise en œuvre d'un procédé tel que décrit précédemment, comprenant : au moins un ordinateur permettant l'exécution d'un module de préparation mettant en œuvre les étapes de préparation des données et d'un module de rendu; et un système de gestion de base de données stockant les valeurs de métadonnées.

La phase du procédé correspondant au mécanisme de préparation est un processus interruptible, car le traitement de chaque ensemble de la donnée native peut être différé dans le temps, l'étape de concaténation se charge de regrouper les ensembles.

Le mécanisme de préparation constitue un traitement pouvant être réalisé par lots dans le cas où les données natives comportent un ensemble de groupe d'objets. Chaque groupe peut être considéré séparément et le traitement de chaque groupe est indépendant, ces dispositions permettant de préparer la donnée avec une taille mémoire limitée et de mettre en place un processus utilisant un chargement partiel en mémoire vive,

Le mécanisme de préparation est un processus automatique, car les algorithmes de simplification sont choisis de manière à limiter au maximum les réglages des paramètres de simplification,

Enfin, le mécanisme de préparation est un traitement pouvant être réalisé de façon multitâche. En effet, suite à une première étape de division spatiale en métavoxels des groupes d'objets, chaque métavoxel est traité de manière indépendante, ce qui permet de réduire le temps de préparation de la base de données.

Selon un aspect de l'invention, chaque métavoxel comprend un ensemble de triangles, les triangles intersectés des objets sont stockés de manière contiguë en conservant une métadonnée correspondant à un identifiant de l'objet pour les différencier.

Selon un autre aspect de l'invention, les données géométriques de la scène sont séparées en groupes, chaque groupe étant traité indépendamment lors de l'étape de formation des métavoxels.

Selon un autre aspect de l'invention, lors de la phase de préparation, les groupes d'objets sont concaténés ou fusionnés, les données géométriques étant stockées de manière contigüe pour chaque métavoxel en conservant une métadonnée sous forme d'un identifiant de groupes et d'objets.

Selon un autre aspect de l'invention, les identifiants de groupe et d'objets sont concaténés.

Selon un autre aspect de l'invention, des moyens d'accélération graphique matériels sont utilisés pour procéder à un rendu en associant une clé à un code couleur dans un tampon de rendu permettant d'identifier les clés dans ce tampon.

Selon un autre aspect de l'invention, une grille régulière récursive est utilisée qui comporte au moins deux niveaux de définitions de grille régulière, les clés d'indexation des métadonnées étant fractionnées et stockées de façon réparties sur les différents niveaux de grille.

Selon un autre aspect de l'invention, des moyens d'accélération graphique matériels ne sont pas utilisés, la clé est reconstruite lors du rendu du voxel en réunissant les parties de la clé associées aux charges utiles associées aux voxels des différents niveaux de définition de la grille lors du rendu, en interrogeant la charge utile du voxel traversé par le rayon.

Selon un autre aspect de l'invention, les objets de la scène 3D sont rassemblés en groupes, et lors de la sélection d'un élément de la scène, il est possible à partir du métavoxel de plus faible niveau d'obtenir une information sur les groupes d'objets présents dans ledit métavoxel, puis de croiser cette information avec les identifiants des objets stockées au niveau du métavoxel de niveau supérieur pour limiter la sélection d'objets possibles.

La présente invention concerne également un système destiné à la mise en œuvre d'un procédé tel que décrit ci-dessus, comprenant :

- au moins un ordinateur permettant l'exécution d'un module de préparation mettant en œuvre les étapes de préparation des données et d'un module de rendu; et
- un système de gestion de base de données stockant les valeurs de métadonnées.

Brève description des dessins

L'invention sera mieux comprise à l'aide de la description détaillée qui est exposée ci-dessous en regard du dessin annexé dans lequel :

La figure 1 est un schéma d'ensemble d'une architecture matérielle permettant de mettre en œuvre le procédé selon l'invention.

La figure 2 est un organigramme de la partie de préparation des données d'un procédé selon l'invention, limitée aux métadonnées et aux métavoxels.

Les figures 3 à 5 illustrent les étapes de métavoxelisation et de voxélisation du procédé de figure 2 ;

Les figures 6, 7 et 8 illustrent des étapes de simplification de la géométrie du procédé de figure 2 ;

La figure 9 illustre l'organisation du stockage des données préparées suite aux étapes illustrées sur la figure 2 ;

Les figures 10 à 14 illustrent des détails du stockage des données destiné à une utilisation dans le second mode de rendu sans carte graphique.

La figure 15 illustre le calcul d'une représentation de voxel sous forme de plan.

La figure 16 illustre le calcul d'une représentation de voxel sous forme de primitive.

La figure 17 illustre l'organisation des éléments 3D sous forme d'objet et groupes.

La figure 18 illustre les étapes du procédé selon l'invention relative à au mode de rendu avec carte graphique.

La figure 19 illustre l'étape de calcul des métavoxels visibles depuis la caméra mentionnée sur la figure 18.

La figure 20 illustre l'étape de calcul des niveaux de détails mentionnée sur la figure 18.

La figure 21 illustre les étapes du procédé selon l'invention relatifs au second mode de rendu sans carte graphique.

La figure 22 illustre l'étape de détermination du mode de rendu mentionnée sur la figure 21.

La figure 23 illustre l'étape de calcul du rendu mentionnée sur la figure 21.

La figure 24 est un tableau illustrant un relevé de performance du procédé sur une plateforme de test pour le second mode de rendu avec carte graphique.

Description détaillée de modes de réalisation particuliers

I. Définitions

Le terme « métavoxel » désigne une subdivision en volume de la scène et la structure de donnée correspondant à cette subdivision.

Un métavoxel peut comprendre notamment comprendre :

- un sous-ensemble de données 3D, ou
- une représentation de ces données sous forme d'un ensemble de voxel, ou encore
- une nouvelle subdivision en métavoxels.

Dans la description ultérieure, le terme métavoxel est employé pour désigner un premier type de métavoxel consistant une subdivision du volume de la scène comprenant un sous-ensemble de données 3D, constitués notamment par des triangles, et un second type de métavoxel, consistant en une subdivision du volume de la scène, et comprenant soit des voxels, soit une nouvelle subdivision en métavoxel.

Le terme « voxel » désigne un élément de volume unitaire de la scène pouvant être plein ou vide, ce qui correspond à la présence ou non de matière.

Un voxel plein correspond également à une représentation des données 3D natives comprises dans cet élément de volume lors du rendu sous forme d'un cube, d'un plan ou d'une normale ou d'une primitive, ou encore d'un « niveau virtuel » comprenant une subdivision virtuelle en plusieurs cubes, comme cela sera détaillé ci-dessous.

Le terme « groupe d'objet » ou groupe est relatif à des regroupements de données géométriques natives. Les données natives comportent usuellement un ensemble de groupe d'objets. Par exemple, dans le cas d'un ensemble de données natives représentant une voiture, cet ensemble comprend une pluralité de groupes, qui peuvent représenter par exemple l'échappement, le moteur, la carrosserie, l'habitacle, chaque groupe comprenant des objets, comme par exemple un pot d'échappement, un filtre catalytique, un volant, une soupape, une durite. Le regroupement des données natives en groupe n'est toutefois pas obligatoire.

Le terme « métadonnées » concerne des données non géométriques comme des propriétés, des données sur les matériaux, l'historique de création, le producteur et la date de génération d'un élément de

la géométrie, mais aussi des informations d'appartenance d'un élément de la géométrie à un objet ou à un groupe d'objet.

II. Architecture du système

Nous allons à présent décrire un procédé et un système selon l'invention en référence aux figures. Ainsi, selon un mode de réalisation représenté sur la figure 1, un système selon l'invention comprend un serveur 2 hébergeant un premier module de préparation des données Prep, destiné à transformer les données natives Nat dans une structure de données optimisée OD par rapport aux données numériques natives, selon un traitement automatisé utilisant un chargement partiel en mémoire vive, et un second module de rendu R1 ou R2, respectivement avec et sans carte graphique. Ces modules R1 et R2 opèrent le rendu sur la base de la structure de donnée optimisée OD.

L'objectif du mécanisme de préparation des données est de réaliser une préparation des données le plus rapidement possible, en utilisant un traitement multitâche par lots, de façon automatique, et en réduisant le nombre de paramètres de traitement pour obtenir des données optimisées stockées de façon compactes et organisées de manière efficace.

La préparation des données fournit deux types de données DB1 et DB2 adaptées aux deux modes de rendus distincts.

Le rendu est effectué sur différents types de matériels hétérogènes, soit en réalisant un rendu R1 utilisant les moyens d'accélération graphique (GPU) et le ou les processeurs génériques (CPU) dans le cas ou des moyens d'accélération graphique sont présents sur le matériel 3, constitué par exemple par un ordinateur personnel destiné à une utilisation pour des logiciels 3D, soit en rendu en utilisant uniquement le ou les processeurs génériques (CPU) d'un matériel 4 ne comprenant aucun moyen d'accélération graphique, ce type de matériel correspondant par exemple à un ordinateur de type portable destiné à des opérations de bureautique ou de consultation d'Internet.

Ces modules de rendu R1 ou R2 assurent un rendu performant, avec le meilleur niveau de détail possible. Dans tous les cas un affichage à l'écran est réalisé, cet affichage pouvant être dégradé. Le rendu utilise un chargement partiel en mémoire vive et prend en compte un budget mémoire.

Il est à noter qu'il est possible de considérer pour le module de préparation Prep de données d'autres sources de données, en utilisant un connecteur C opérant des transformations de données.

Des moyens de protection et de garantie de la confidentialité des données peuvent également être mis en place de façon optionnelle.

Par ailleurs, il est également possibilité d'utiliser d'autres moteurs de rendu (GPU/CPU) à l'aide de connecteurs C.

Les modules de préparation Prep et de rendu R1 ou R2 sont représentés sur la figure 1 de façon répartie sur différentes plateformes matérielles. Il est possible dans ce cas de mettre en place un mécanisme de réplication des données DB1 ou DB2 à destination des plateformes matérielles 3 et 4 sur lesquels sont exécutés les opérations de rendu.

De façon alternative, il est possible d'exécuter le module de rendu R1 ou R2 sur la même plateforme matérielle que le module de préparation Prep.

III. Préparation des données

Nous allons à présent décrire les étapes de préparation des données en référence aux figures 2 à 9.

La figure 2 fait apparaître à la fois un traitement des métadonnées MD et des données géométriques D présentent dans les données natives Nat.

L'objectif des étapes de préparation par le module de rendu Prep est d'obtenir une organisation compacte et efficace pour un rendu hors du noyau.

Ainsi, en ce qui concerne les métadonnées MD, après une première étape EP1 de conversion des métadonnées, une étape EP2 de création d'un dictionnaire de métadonnées est effectuée de façon à séparer les métadonnées et à obtenir des couples clé K/valeur V.

Un stockage différencié est opéré. Les clés K sont injectées dans la géométrie dans une étape EP4.

Les valeurs V sont stockées hors de la géométrie en particulier dans un système de gestion de base de données (SGBD).

Ces dispositions permettent d'assurer une grande compacité des données, l'ensemble des clés K prend peu de place et peut être détaché de l'ensemble des valeurs V.

L'intérêt de cette approche est de permettre une interaction par métadonnées (groupement, mettre en évidence/cacher des objets, identification des objets).

Les clés K sont injectées dans la géométrie D sous forme d'attributs de données non géométrique.

Par exemple, ces clés peuvent être introduites dans les données des triangles en tant qu'attributs de sommets, ou encore dans les données de voxel, en tant que champ de bits dans la charge du voxel.

Les clés sont générées de manière automatique.

Dans une étape EP3, il est possible de réaliser une réplication de la base de données des métadonnées sur un poste client 3, 4, afin de faciliter le déploiement des métadonnées sur le poste client.

Il est en fait possible de prévoir deux types de bases de métadonnées :

- un système de gestion de base de données (SGBD) multi utilisateur déployé sur un serveur, par exemple sur le serveur sur lequel est exécuté le module de préparation Prep, ce SGBD étant attaqué par plusieurs sous serveurs de préparation distants, et

- un SGBD mono utilisateur léger, par exemple sur un poste client 3, 4 ou s'exécute un module de rendu R1 ou R2.

L'étape de réplication correspond à une extraction des données nécessaire du premier SGBD pour générer une base plus petite dans le second SGBD.

En ce qui concerne les données géométriques, une première étape EP5 de conversion à partir des données natives est réalisée. Les données géométriques D sont des représentations numériques constituées par des maillages, surfaces implicites, ou d'autres représentations par exemple de type brep. L'étape de conversion consiste à transformer, en utilisant des connecteurs ou adapteurs les représentations diverses des données natives Nat en une représentation commune attendue par les étapes ultérieures du procédé, en assurant une fidélité maximale des données

Une fois les données converties dans un format commun, une étape de métavoxelisation EP6 est réalisée.

La métavoxelisation consiste à associer les éléments de donnée numérique D, constitués notamment par des triangles, aux cubes d'une grille

régulière divisant l'espace dans lequel est plongée la scène 3D dont le rendu doit être effectué.

La métavoxelisation comprend une étape préliminaire de calcul des boîtes englobantes (BE) des objets de la scène. Ces boîtes sont alignées sur les axes d'un repère AABB (Axis Aligned Bounding Box) (Figure 3). Ces boîtes englobantes sont calculées dans le repère local de l'objet, et la position de l'objet dans le repère global de la scène est importée depuis les métadonnées.

Cette étape préliminaire est réalisée de préférence lors de l'importation des métadonnées.

Par la suite, les boîtes englobantes sont exprimées dans le repère commun ou global.

La métavoxelisation comprend ensuite :

- une étape de précalcul consistant à tester l'intersection des boîtes englobantes des objets de la scène avec les cellules de la grilles, et
- une étape consistant à tester les intersections entre les triangles des objets dont la boîte englobante forme une intersection avec ladite cellule.

Ces dispositions permettent de diminuer fortement le temps de calcul de la métavoxelisation par rapport à un test naïf d'intersection de tous les triangles des objets de la scène avec l'ensemble des cellules de la grille.

Ainsi, le test naïf occasionne un coût de calcul proportionnel à n*p*k où n représente le nombre de cellules de la grille, p le nombre d'objets de la scène et k le nombre de triangles moyen par objet, alors que la méthode en deux sous-étapes décrites ci-dessus occasionne un coût de calcul proportionnel à n*p+m*k où m représente le nombre de cellules contenant potentiellement des éléments de la géométrie, ce nombre étant en général bien inférieur à n.

Lors de l'étape de précalcul, une liste Lcell associe chaque élément de géométrie à un ensemble de cellules de la grille contenant de la géométrie. Cette liste est constituée en insérant pour chaque objet de la scène, les cellules de la grille présentant une intersection avec sa boîte englobante. Une fois la liste Lcell construite, un parcours de cette liste permet de construire, une liste Lgeom qui pour chaque cellule, associe les objets présentant une intersection ladite cellule est constituée.

A titre d'exemple, la figure 3 représente quatre cellules CG0, CG1, CG2, CG3 de la grille et deux objets O1 et O2. La boîte englobante BE(O1) de l'objet O1 présente une intersection avec les cellules CG0 et CG2, et la boîte

englobante BE(O2) présente une intersection avec les cellules CG0 et CG1. Dans cet exemple, on obtient dans ce cas :

Lcell[O1]={CG0, CG2}

Lcell[O2]={CG0, CG1}

Les listes Lgeom sont ensuite construites en croisant les valeurs des listes Lcell.

Lgeom[CG0]={O1,O2}

Lgeom[CG1]={O2}

Lgeom[CG2]={O1}

Lgeom[CG3]={}

Lors de l'étape suivante de gridding, les intersections des triangles des objets avec les cellules de la grille sont réalisées. Dans un mode de réalisation particulier, pour chaque cellule CG, la liste des objets intersectés Lgeom est considérée, et pour chaque objet de cette liste, un test d'intersection est réalisé pour chaque triangle T de l'objet afin de déterminer si le centre de gravité de ce triangle est compris dans la cellule CG.

Cette disposition basée sur l'utilisation de la position du centre de gravité permet une résolution univoque de l'appartenance entre une cellule et un triangle : un triangle appartient à une seule cellule de la grille.

A titre d'exemple, la figure 4 représente deux cellules CG0 et CG1, et un objet s'entendant sur ces deux cellules et comprenant deux triangles T1 et T2.

Le centre de gravité du triangle T1 étant situé dans la cellule CG0 et le triangle T2 étant situé dans la cellule CG1, le triangle T1 est rattaché à la cellule CG0 alors que le triangle T2 est rattaché à la cellule CG1, alors que ces deux triangles présentent chacun des intersections avec les deux cellules.

Des métavoxels, c'est-à-dire des sous-ensembles de données 3D appartenant à une cellule de la grille, constitués notamment par des triangles, sont ainsi constitués.

Chaque métavoxel comprend un ensemble de triangles, les triangles intersectés des objets sont stockés de manière contigüe en conservant une métadonnée M correspondant à un identifiant de l'objet pour les différencier.

Une liste de triplets d'indices et une liste de sommets stockés de manière contigües sont ainsi obtenues.

Dans le cas du premier type de rendu utilisant une accélération graphique par GPU, ces dispositions permettent :

- un chargement depuis le stockage sur disque (HDD) vers la mémoire vive (RAM) du microprocesseur (CPU), puis dans la mémoire vidéo ou VRAM du processeur graphique (GPU) ; et
- un rendu rapide sur GPU en minimisant les changements d'états OpenGL et en s'adaptant à l'architecture des GPU).

Ainsi, dans le cas représenté sur la figure 5 d'un premier objet O1 comprenant deux triangles, le premier triangle étant formé comprenant trois sommets correspondant aux positions P0, P1, P2, le second triangle étant formé par les sommets correspondant aux positions P1, P2 et P3, et d'un deuxième objet O2 comprenant un triangle formé par les sommets aux positions P4, P5, P6, la liste des sommets LS et la liste des indices LI peuvent être stockées de la façon suivante :

LS = {(P0; M(O1)) (P1;M(O1)) (P2;M(O1)) (P3;M(O1)) (P4; M(O2) (P5;M(O2)) (P6;M(O2))}.

LI = (0;1;2) (1;2;3) (4;5;6)…

Dans lequel M correspond à une métadonnée représentant l'identifiant d'un objet, P une position, et les valeurs des indices dans la liste des indices correspondent aux rangs des sommets dans la liste LS.

Il est à noter que dans le cas du premier type de rendu R1 utilisant une accélération graphique, les clés K des métadonnées permettent de retrouver les valeurs dans le dictionnaire de métadonnées et de différencier les éléments d'un même métavoxel appartenant à des objets différents.

Le stockage peut être réalisé en tant qu'attribut de sommet. Un sommet peut contenir en effet des données de :

- Position : xyz sur 3*16bits
- Normale : xyz sur 3*16bits
- des attributs métadonnée sous formes de clés sur 3*8 bits
- des coordonnées de textures
- d'autres attributs génériques

Le fait de stocker les métadonnées dans la géométrie permet un gain mémoire car le stockage peut être réalisé dans la mémoire vidéo (VRAM). Il n'est donc pas nécessaire de stocker les clés K dans la mémoire vive (RAM). Les données D de géométrie peuvent alors être supprimées de la mémoire vive (RAM) une fois chargées dans la mémoire vidéo (VRAM).

Après l'étape de métavoxelisation, les étapes ultérieures de préparation sont distinctes pour la préparation destinée au premier mode de rendu (avec GPU) et au second mode de rendu (sans GPU).

1. Préparation des données – étapes relatives au premier mode de rendu

Dans le cas d'une préparation pour le premier mode de rendu, une étape EP7 de simplification est réalisée.

Lors de l'étape de simplification, le but est de générer pour les données géométriques contenues dans chaque métavoxel plusieurs niveaux de détail (LOD) de façon à pouvoir afficher ces données géométriques natives avec une mémoire vive (RAM) limitée.

En particulier, cinq niveaux de détails peuvent être générés et stockés pour chaque métavoxel :

- un niveau n0 correspondant aux données natives, en utilisant simplement le résultat de l'étape de métavoxelisation et le stockage contigu de ces données ;

- un niveau n1 dans lequel un évidage de la géométrie est réalisé : seule la surface externe est conservée (Figure 6);

- un niveau n2 utilisant une simplification légère assurée par la fusion de sommets (Figure 7);

- un niveau n3 utilisant une simplification importante également assurée par la fusion de sommets ;

- un dernier niveau n4 utilisant des « imposteurs », c'est-à-dire 6 vues du contenu du métavoxel, sous formes de textures plaquées sur des quadrilatères (Figure 8).

Les niveaux n1 à n4 sont détaillés ci-dessous.

Dans le cas d'un évidage correspondant au niveau n1, comme cela est représenté de façon schématique sur la figure 6, seule l'enveloppe externe O1ext des objets est conservée, les éléments intérieurs O1int étant supprimés. A cet effet, quatre étapes sont mises en œuvre : dans un premier temps, une discrétisation des éléments de géométrie dans une grille de voxels est réalisée. Puis un remplissage des voxels extérieurs Vext est réalisé. On détermine ensuite la frontière, c'est-à-dire des voxels à la zone limite remplie/non remplie. Le maillage évidé est ensuite généré en extrayant les triangles appartenant à cette frontière.

Dans le cas d'une simplification correspondant au niveau n2 ou n3, l'objectif est de simplifier l'enveloppe externe des objets. Deux étapes sont mises en œuvres, comme illustré sur la figure 7 : le maillage est plongé dans une grille G, pour réduire le temps de calcul, puis, pour chaque cube de la grille et chaque sommet s, dans ce cube, la distance entre deux sommets est comparée à une limite d déterminée. Si la distance est inférieure à d, les sommets sont fusionnés. Dans la figure les ensembles de sommets {s1;s2} et {s3;s4} sont fusionnés. Il est à noter que la limite d peut être fixée automatiquement par exemple comme une fraction du rayon de la sphère englobante du maillage.

Dans le cas d'une génération d'imposteurs correspondant au niveau n4, un rendu est réalisé pour un nombre limité nv de points de vue, par exemple 6 (devant, derrière, haut, bas, gauche, droite), puis le résultat du rendu est capturé dans une texture, puis l'imposteur I est généré sous forme de nv quadrilatères sur lesquels sont plaqués les nv textures.

Suite à l'étape de simplification, une étape de concaténation et de compression EP8 est réalisée. L'objectif de ce cette étape est de regrouper les données en un ou plusieurs méta-fichiers, de faciliter le déploiement des données, et notamment d'accélérer la copie de lors de l'installation, et d'accélérer le rendu en optimisant les transferts entre le disque (HDD) et la mémoire vive (RAM) par une utilisation du cache disque (HDD) et un mécanisme de transfert entre le disque (HDD) et la mémoire vive (RAM) du système d'exploitation.

Dans le cas du premier mode de rendu (avec GPU), le stockage est réalisé sous forme de deux ensembles, comme cela est illustré sur la figure 9 : un premier ensemble comprenant des index, et un second ensemble comprenant les données.

Chaque index mvID est un identifiant unique qui correspond à un métavoxel et permet d'accéder à celui-ci. Dans le premier ensemble, la taille décompressée T de chaque métavoxel est également stockée, pour les différents niveaux de détails, de façon à pouvoir prévoir le budget mémoire lors du rendu, ce qui sera détaillé ultérieurement.

Les données sont stockées dans un second ensemble de manière séquentielle, séparées en bloc Bk de la taille d'un cluster d'un multiple de 512

Les données sont stockées compressées et sont décompressées à la volée.

Les données contiennent les Métavoxels stockés par niveaux de détail n0, n1, n2, n3, n4.

2. Préparation des données – étapes relatives au second mode de rendu

Nous allons à présent décrire les étapes ultérieures à la métavoxelisation EP6 relatives à la préparation d'un rendu selon le second mode de rendu (sans GPU).

Suite à l'étape de métavoxelisation, une étape de voxelisation EP9 est réalisée.

Ainsi, pour chaque métavoxel obtenu lors de l'étape EP6, une subdivision en métavoxel de niveau supérieur de détail est réalisée. Chaque métavoxel de niveau supérieur de détail peut être à son tour divisé en un ensemble de métavoxel de niveau supérieur de détail.

Comme cela est représenté sur la figure 11, une grille régulière récursive est utilisée qui comporte plusieurs niveaux de définitions de grille régulière.

Un métavoxel d'un niveau donné de la grille correspond à une subdivision d'un métavoxel de niveau inférieur de détail.

A titre d'exemple, comme cela est représenté sur la figure 11 pour chaque métavoxel de niveau i, une grille peut être définie dans laquelle le métavoxel de niveau i est divisé ni fois par axe, par exemple avec ni=16.

Les éléments de volume ou voxels obtenus de niveau i peuvent à leur tour constituer un métavoxel de niveau i+1 pour lequel un nouveau niveau de grille peut être définie en procédant à une nouvelle division ni+1 fois par axe avec par exemple ni+1=16.

Comme cela est décrit sur la figure 11, pour faciliter le parcours dans la grille de chaque niveau, des divisions intermédiaires ou un niveau intermédiaire i.0 peuvent être définis. Ces niveaux permettent de définir une première subdivision intermédiaire du métavoxel ni0 division par axe, avec par exemple ni0=4, puis une seconde subdivision intermédiaire ni1, avec par exemple ni1=4 de façon à obtenir un voxel de niveau i, Le nombre de subdivision sur chaque axe ni est égal au produit de ni0*ni1.

Il est bien entendu possible de choisir des nombres de division ni différents pour chaque niveau de grille.

La figure 12 illustre la structure d'un identifiant unique de métavoxel métavoxelID, dans le cas ou quatre niveaux de grille sont utilisés.

Les quatre niveaux correspondent à :

- un niveau de base correspondant à la division réalisée dans le cadre de la métavoxélisation commune au premier mode de rendu (utilisant un GPU) et second mode de rendu (sans GPU), et

- trois niveaux spécifiques au second mode de rendu.

Cet identifiant comprend :

- une information depth sur la profondeur du métavoxel considéré dans les niveaux de profondeur de la grille régulière ;

- des informations de position dans les différents niveaux de grille et en particulier :

- l'identifiant ID Scene du métavoxel du niveau de base correspondant à la position selon trois axes XYZ par rapport au repère global de la scène, cette position étant repérée par des indices de positionnement dans la grille (position identique au métavoxel correspondant de l'étape EP6);

- la position Pos 0, Pos 1, Pos. 2, dans les niveaux de grille spécifiques au rendu 2D.

Il est à noter que les positions Pos 0, Pos 1, Pos. 2 peuvent être décomposés en deux indices Lvl0iD et Lvl1iD correspondant aux niveaux intermédiaires de la grille à chaque niveau.

Comme cela est illustré par la figure 12, l'identifiant Méta Voxel ID est un identifiant unique, utilisé pour tous les métavoxels des différents niveaux de profondeur de la grille régulière.

Bien entendu, un nombre de niveaux de grille supérieur ou inférieur à quatre peut être utilisé.

Nous allons à présent détailler la réalisation de la voxelisation permettant de définir pour chaque métavoxel de chaque niveau de la grille des représentations sous forme de voxel ou cube, de plan ou de primitive ou encore de « niveau virtuel ».

Un voxel correspond à la présence ou non de matière : au rendu, ce volume peut faire l'objet de plusieurs représentations.

Selon une première possibilité, les voxels pleins peuvent être représentés par des cubes.

Pour améliorer la qualité de rendu, il est possible d'associer une normale à chaque voxel qui permet de simuler une surface pour le calcul de l'éclairage.

Dans le cas d'un « niveau virtuel », le volume du voxel est représenté par une subdivision en des cubes correspondant à une subdivision du volume du voxel en 2 sur chaque axe. Ces cubes peuvent être pleins ou vides. Par exemple 8 cubes sont définis. Ces cubes partagent une même définition de la normale qui correspond à la normale du voxel père.

L'utilisation de ces cubes permet de raffiner la représentation en volume sans nécessiter un niveau de grille régulière supplémentaire.

Il est également possible d'utiliser une représentation par un plan et une normale.

Pour améliorer l'aspect, le cube peut être remplacé par un ensemble fini de configurations géométriques encore appelées primitives.

Nous allons à présent détailler des méthodes de calcul des représentations en volume à partir des données 3D représentées sous formes de triangles.

Dans le cas d'une représentation en cube du voxel, il suffit d'identifier les voxels contenant de la matière et de les représenter par un voxel plein.

Pour calculer une normale correspondante, il est possible par exemple de calculer une normale correspondant à la moyenne des normales des triangles contenus dans le volume destinés à être représentés par le voxel.

Comme cela est représenté sur la figure 15, dans le cas d'une représentation par un plan et une normale, la première étape consiste à calculer une normale représentant le mieux le voxel, par exemple comme une normale moyenne sur les triangles de la géométrie, puis à projeter tous les sommets A de la géométrie en un point B sur la droite définit par la normale et le centre du voxel.

Il est ensuite possible de définir deux positions extrêmes des projections Min et Max sur la droite et de définir un premier et un second plan perpendiculaires à la normale passant par ces deux positions extrêmes. La représentation du voxel correspond aux deux plans ainsi définis.

Dans le cas de l'utilisation de primitives, il est possible, comme cela est illustré sur la figure 16, de définir par exemple :

(A) – Un premier type de primitives à partir de 6 normales correspondant à des plans coupant le cube en diagonale, définissant ainsi 12 demi-cubes prismatiques en tant que primitives de part et d'autre de chaque plan,

(B) - Un second type de primitives à partir de 3 normales correspondant à des plans coupant le cube en 2, parallèlement à ses côtés, définissant ainsi 6 pavés en tant que primitives et

(C)- Un troisième type de primitives à partir de 8 subdivisions du cube en tant que cubes présentant une arête de dimension divisée par 2, ce qui correspond à la définition de 8 primitives.

Pour les deux premiers types de primitives, il est possible de déterminer si une primitive est utilisable en :

- réalisant une projection des vertex de la géométrie sur la normale, puis

- à déterminer si tous les points projetés se trouvent dans le même demi espace.

Si tous les points projetés sont dans le même demi-espace, alors la forme peut être retenue. Il est possible de retenir arbitrairement la première qui satisfait le critère.

Si aucune des primitives des deux premiers types ne convient, un test est réalisé pour identifier si tous les triangles sont contenus dans un seul cube présentant une arête correspondant à la moitié du voxel conformément au troisième type de primitive.

Si aucune des primitives ne convient, on utilise la représentation standard d'un voxel sous forme d'un cube plein de la taille du voxel.

La voxélisation est réalisée de façon indépendante par les différents threads ou processus pour chaque métavoxel dans la grille à plusieurs niveaux, en partant de la racine et en parcourant l'arborescence des voxels.

La voxelisation étant réalisée de façon indépendante pour les métavoxels, et les résultats étant stockés de manière séquentielle, une seconde étape de réorganisation permet de rassembler de façon contiguë les données relatives à l'ensemble des voxels fils d'un même voxel père, ce qui correspond à une réorganisation des voxels. Ces dispositions permettent d'améliorer les performances lors du rendu, et notamment les interrogations de la grille régulière récursive.

Pour chaque métavoxel, une fusion est opérée avec les métavoxels voisins. La voxelisation est effectuée sur ce groupe de métavoxels, de façon indépendante en utilisant une approche multiprocessus et multithreads pour la voxelisation.

Cette disposition a pour objet d'éviter des trous éventuels lors de la voxelisation causés par le fait que des triangles se trouvant sur la frontière entre deux métavoxels sont affectés de façon univoque à l'un de ces deux métavoxels en fonction de la position de leur centre de gravité, comme cela a été décrit précédemment.

Le stockage est réalisé sous forme de deux ensembles, comme cela est illustré sur la figure 10 : un premier ensemble comprenant des index, et un second ensemble comprenant les données. Le but est d'optimiser les transferts vers le disque (HDD) par un stockage séquentiel mais en permettant un accès direct.

Nous avons précédemment détaillé la structure des index qui correspondent aux identifiants metavoxelID en référence à la figure 12. Nous allons à présent décrire le stockage des données.

Les données sont stockées sous forme de blocs.

Les blocs sont organisés de la façon suivante.

Les blocs comprennent un entête et des charges utiles des voxels.

L'utilisation d'un entête permet une économie d'espace de stockage et un accès direct aux données.

La structure de stockage des données relatives aux métavoxels est illustrée sur la figure 13a.

L'entête comporte une table d'index IndexTbl de mémoire vive (RAM).

Cette table d'index permet d'accéder à l'ensemble de la charge utile dans le cache (RAM) alors qu'elle elle est stockée de façon non contiguë dans ce cache. Le cache RAM est alloué par le procédé pour réaliser le chargement des charges utiles de voxels.

La table d'index comprend :

- un identifiant ID qui correspond au métavoxelID détaillé en référence à la figure 12 ;
- un index vers le premier bloc de mémoire du méta-voxel père ;
- un index de fichier vers le père, en vue de restaurer celui-ci dans la charge utile ;

- des pointeurs vers les autres blocs relatifs au même métavoxel ;
- le nombre de ces blocs ;
- un compteur d'usage correspondant aux nombre de threads ayant accès aux données à un instant donné, afin d'identifier si celle-ci peuvent être ou non supprimées du cache ;
- un remplissage permettant de s'assurer que les champs de charge utile ne s'étendent pas sur deux blocs.

L'entête comporte également un masque de présence qui permet d'indiquer pour chaque voxel si celui-ci est vide ou plein, et ceci pour chaque niveau de profondeur de la grille. Par exemple, le masque peut comprendre un 0 pour indiquer un voxel vide et un 1 pour un voxel plein (c'est-à-dire contenant de la géométrie). Ce masque de présence permet de simplifier les calculs d'intersection. Il n'y ainsi pas besoin d'accéder aux données relatives à chaque voxel.

Selon un aspect particulier, un premier masque de présence Msk Voxel i.0 est stocké pour des voxel représentant une subdivision au niveau intermédiaire de chaque métavoxel comme cela a été détaillé en référence à la figure 11, et un ensemble de masque de présence Msk i.1 est stocké pour chacun de ces voxels de niveau intermédiaire. Une information de décalage est également stockée qui permet de retrouver facilement la charge utile du voxel.

Ces dispositions permettent de faciliter encore les calculs d'intersection de rayon, en permettant de tester dans un premier temps l'intersection avec un voxel intermédiaire, puis de ne tester les intersections qu'avec les voxels compris dans les voxels intermédiaires pour lesquels une intersection est détectée.

La figures 13b permet de décrire la structure des masques et des décalages dans un exemple simplifié en considérant des masques de 4 bits pour les masques i.0 et i.1.

En supposant que le masque i.0 présente le contenu suivant : [1,0,1,1], une table est constituée qui stocke pour chaque entrée du masque i.0 un masque i.1 et un décalage correspondant au nombre de voxels pleins répertoriée avant le voxel intermédiaire i .0 courant.

Ainsi, comme cela est décrit sur la figure 13b pour la première ligne, un masque i.1 est stocké avec par exemple le contenu suivant [0,1, 0, 1] qui comprend donc deux voxel pleins. Un offset ou décalage nul est stocké car il s'agit du premier voxel intermédiaire i.0 traité.

La seconde ligne correspond à une entrée à 0 dans le masque i.0. Dans ces conditions, aucun masque i.1 n'est stockée l'information de décalage n'est pas stockée non plus.

La troisième ligne correspond à une entrée à 1 dans le masque i.0. un masque i.1 est stocké avec par exemple le contenu suivant [1,0, 1, 1] qui comprend donc trois voxels pleins. Un offset ou décalage de 2 fois la taille d'une charge utile PL d'un voxel PlSize, par exemple de 128bit, est stocké correspondant aux deux voxels pleins de la première ligne.

La quatrième ligne correspond à une entrée à 1 dans le masque i.0. un masque i.1 est stocké avec par exemple le contenu suivant [0,0, 1, 1] qui comprend donc deux voxels pleins. Un offset ou décalage de (2+3) fois la taille d'une charge utile d'un voxel PlSize est stocké correspondant aux deux voxels pleins de la première ligne et aux trois voxels pleins de la troisième ligne.

Pour chaque ligne, le décalage correspond à la taille de la charge utile PL d'un voxel multiplié par le nombre cumulé de voxels plein dans les lignes précédentes.

Ainsi, pour calculer l'emplacement de la charge utile PL du voxel correspondant au bit 3 du mask i.1 de la ligne 3 :

- on vérifie que le bit 3 du mask i.0 soit à 1 (si non le voxel que l'on considère est vide) ;
- on récupère ensuite le mask i.1 de la ligne 3 (Lsb[0,0,1,1]Msb),
- on compte le nombre de voxels pleins (nbVoxPlein) avant le voxel considéré. Ici nbVoxPlein vaut 1 car seul le bit 2 est à 1 avant le bit 3.

Comme cela est illustré sur la figure 13c, La position de la charge utile considérée est donnée par la relation suivante.

Offset + nbVoxPlein*PlSize = (2+3)*PlSize + nbVoxPlein*PlSize = 5*PlSize+1*PlSize.

La structure de stockage des données relative aux métavoxels comporte en outre un champ de donnée qui comprend les charges utiles des voxels pleins.

Les charges utiles des voxels sont stockées séquentiellement. La charge utile comprend la description des voxels.

Selon un mode de mise en œuvre, quatre types de charge utiles de voxel sont possibles :

- Les nœuds (Node), voxel intermédiaire qui pointe vers des voxels d'un niveau plus élevé dans l'arborescence;

- Les feuilles pouvant être : soit une représentation par plan/normale, soit une primitive, de type cube ou autre.

La structure des données de charge utile correspondant à ces différents types est illustrée à la figure 14.

Le stockage d'une charge utile comprend :

- une zone utile « Free to Use »,

- une donnée Klm représentant la taille des données décompressées des enfants de cet éléments, la donnée Klm comprenant le nombre K de sous-voxels pleins dans ce voxel, et

- un type de charge (Nœud, primitive, plan/normale, niveau virtuel).

Il est à donner que la donnée Klm permet de faire la prévision du budget mémoire.

Dans le cas d'un nœud, la zone utile comprend un pointeur dans le cache vers un métavoxel de niveau plus défini supérieur, et du remplissage.

Dans le cas d'une primitive, la zone utile comprend :

- la taille compressée du métavoxel de niveau plus défini,

- un index de fichier sur le disque vers le métavoxel de niveau plus défini,

- un espace de stockage pour des identifiants de groupes et d'objets,

- un identifiant du type de primitive, et

- un identifiant d'une normale obtenue à partir de la géométrie native qui est représentée par la primitive. Des valeurs de normales possibles discrétisées prédéfinies sont stockées dans un tableau, L'identifiant de la normale correspond à une entrée dans ce tableau.

Dans le cas d'un plan/ d'une normale, la zone utile comprend :

- la taille compressée du métavoxel de niveau inférieur,

- un index de fichier sur le disque vers le métavoxel de niveau inférieur - un espace de stockage pour des identifiants de groupes et d'objets,

- les coordonnées de la normale.

Dans le cas d'un « niveau virtuel », les données suivantes sont stockées :

- la taille compressée du métavoxel de niveau inférieur,

- un index de fichier sur le disque vers le métavoxel de niveau inférieur - un espace de stockage pour des identifiants de groupes et d'objets,

- un masque de présence indiquant si chacun des cubes compris dans le niveau virtuel est plein ou vide,

- les coordonnées de la normale.

A titre d'exemple, les normales peuvent être discrétisées sur 14 bits, les primitives possibles sont au nombre de 2^5 et les normales possibles discrétisées pour cette primitive au nombre de 2^4.

Les charges utiles sont stockées compressées et sont décompressées à la volée lors du chargement.

Comme nous l'avons vu précédemment les éléments géométriques de la scène 3D sont regroupés en objets, les objets appartenant à des groupes. Nous allons à présent décrire en référence à la figure 17, la prise en compte de l'appartenance des éléments géométriques à un objet ou à un groupe d'objet dans le cadre de la préparation d'un rendu selon le second mode de rendu.

Lors de la voxelisation, les données géométriques appartenant aux objets et groupes d'objets sont concaténées ou fusionnées pour chaque métavoxel pour être ensuite représentée par une seule primitive ou un plan. Pour chaque métavoxel, des métadonnées sont conservées permettant d'identifier les éléments appartenant aux différents groupes et aux différents objets.

En particulier, on définit :

- des identifiants ou clefs KO d'objet ou de combinaison d'objets, ainsi que

- des identifiants ou clefs KG de groupe d'objet et de combinaison de groupes d'objets.

Les identifiants ou clés KG KO sont fractionnées et stockées de façon réparties sur les différents niveaux de grille, afin de prendre en compte la taille limitée de charge utile par voxel.

Ces métadonnées sont stockées dans des tables de la façon suivante :

- une table IDGrTbl des identifiants de groupes KG et/ou de combinaison de groupes est stockée pour tous les métavoxels.

- une table IDObjTbl des identifiants ou clés KO d'objets ou de combinaison d'objets est stockée au niveau des métavoxels de plus faible niveau de détail.

Lors de la sélection d'un élément de la scène, il est possible à partir du métavoxel de plus haut niveau de détail d'obtenir une information sur les groupes d'objets présents dans ledit métavoxel, puis de croiser cette information avec les identifiants des objets stockées au niveau du métavoxel de niveau de détail le plus faible pour limiter la sélection d'objets possibles.

La figure 17 illustre un exemple de groupes et d'objets contenus dans ces groupes. Dans ce cas deux groupes d'objet Gr_1 et Gr_2 sont définis comprenant respectivement des objets O_{11}, O_{12}, O_{13}, O_{14}, O_{15}, O_{16}, et O_{21}, O_{22}.

Un métavoxel MV_0 de niveau 0 comprend les objets O_{11}, O_{12} et O_{21}. Les tables IDGrTbl et IDOTbl pour ce métavoxel MV_0 sont alors reproduites ci-dessous :

IDGrTbl (MV_0) :

KG1 -> {Gr_1}
KG2 -> {Gr_2}
KG3 -> {KG1, KG2}

IDOTbl (MV_0) :

KO1-> {O_{11}}
KO2-> {O_{12}}
KO3-> {O_{21}}

Pour le métavoxel MV_1 de niveau 1, seuls des identifiants de groupes d'objets ou de combinaisons de groupes d'objets sont stockés. Ainsi, pour le métavoxel MV_1, seule la clé KG1 sera stockée car le métavoxel MV_1 ne contient que des objets du groupe Gr_1.

IV. Etapes de Rendu

Nous allons à présent détailler la réalisation du rendu selon les deux modes de réalisation évoqués précédemment.

1. Etapes de rendu - Premier mode de rendu

Comme nous l'avons vu précédemment, le premier mode de rendu R1 utilise les moyens d'accélération graphique (GPU) et le ou les processeurs génériques (CPU)

L'objectif étant de réaliser un rendu le plus rapidement possible en termes de temps de chargement et de fréquence d'affichage, tout en respectant le budget mémoire vive (RAM) et vidéo (VRAM) et en maximisant la qualité de rendu correspondant aux niveaux de détail.

Les étapes de rendu sont représentées sur la figure 18.

En ce qui concerne l'utilisation des métadonnées, comme nous l'avons vu précédemment, un dictionnaire couples (clé K, valeur V) a été constitué, que les clés ont été injectées dans la géométrie, et que les valeurs stockées dans un SGDB.

Le traitement d'une requête relative aux métadonnées est traité de la façon suivante.

La clé K est récupérée depuis l'affichage 3D. Ainsi, lors d'une requête de l'utilisateur ER1_0, pour une métadonnée correspondant à un point de l'image, le GPU procède à un rendu dans un mode d'affichage qui permet d'associer une clé à un code couleur. Le tampon de rendu correspondant est alors récupéré sur le CPU qui identifie les clés K dans ce tampon.

Le dictionnaire est alors interrogé à l'aide de la clé K dans une étape ER1_9.

En réponse à la requête ER1_10, un retour visuel sur l'affichage 3D peut être effectué dans une étape ER1_8, par exemple sous forme de mise en évidence ou d'occultation d'un objet, ou encore d'affichage des propriétés, de filtrage des données.

En ce qui concerne la réalisation du rendu en lui-même, il apparaît qu'en réponse à une mise à jour de la caméra dans une étape ER1_1 suite à une interaction utilisateur ER1_0, les chargements des données entre le disque, la mémoire vive (RAM) puis la RAM et la mémoire vidéo (VRAM) sont réalisés.

En ce qui concerne l'ordonnancement de ces tâches, il apparaît que deux tâches de chargement doivent être réalisées, à savoir :

- un chargement entre le disque (HDD) et la mémoire vive (RAM) ; et

- un chargement entre la mémoire vive (RAM) et la mémoire vidéo (VRAM) ;

Dans certaines configurations, par exemple des configurations utilisant OpenGL 2.0, il n'est pas possible de multithreader le chargement entre la RAM et la VRAM. Ainsi, afin de réduire le temps de chargement, les deux tâches de chargement (HDD->RAM et RAM->VRAM) sont séparées et entrelacées. Le chargement HDD->RAM s'effectue sur un thread secondaire et le chargement RAM->VRAM s'effectue sur un thread principal.

La synchronisation entre les deux threads de chargement s'effectue à l'aide de queue de chargement et de déchargement. Ces queues permettent de fragmenter les tâches de chargement afin d'éviter une latence trop longue. Ainsi, plutôt que de charger n metavoxels, p.q métavoxels sont chargés où p est le nombre de metavoxels chargés lors d'une demande et q=n/l, l étant la taille de la queue.

Le traitement des queues de déchargement est également découpé en deux parties avec :

- un traitement des requêtes de mise à jour/chargement sur un thread secondaire, et

- un traitement effectif de la requête sur un thread principal

Les requêtes de chargement ou de mise à jour sont déterminées en réalisant dans une étape ER1_2 un calcul de l'ensemble des métavoxels visibles depuis la caméra puis en calculant les niveaux de détail pour chacun de ces métavoxels dans une étape ER1_3.

L'objectif de l'étape ER1_2 de calcul des métavoxels visibles 11 a pour objectif d'éliminer le plus possible de métavoxels non visibles, c'est-à-dire, comme cela est illustré sur la figure 19 : les métavoxels 10 non contenus dans le frustum de la caméra, représentés hachurés et les métavoxels 12 cachés par d'autres metavoxels, représentés remplis par des pointillés.

En particulier, ce calcul est réalisé en 4 étapes selon un algorithme de type décrit dans le document « Dean Sekulic Chapter 29. Efficient Occlusion Culling, GPU Gems, Nvidia », à savoir :

- Un calcul de l'ensemble des éléments occludeurs inclus dans le frustum caméra ;

- un rendu à une première image t des métavoxels constituant des éléments occludeurs sous forme d'imposteurs I ;

- à l'image t+1 une requête d'occlusion est effectuée sur la base de la première image afin de déterminer si un élément occludeur est visible par rapport aux autres éléments occludeurs du point de vue de la caméra ;

- à l'image t+2 : le chargement des métavoxels ignore les métavoxels cachés.

Différencier les étapes de rendu des éléments occludeurs des requêtes d'occlusion permet de lisser la latence introduite par ces requêtes d'occlusions.

L'étape ER1_3 de calcul des niveaux de détail a pour objectif de déterminer le niveau de détail de chaque métavoxel pour obtenir le meilleur affichage et respecter les contraintes mémoires.

Ce calcul est réalisé en simulant un chargement des métavoxels sur un ensemble de secteurs correspondant à des zones à une distance données de la caméra, puis, pour chaque secteur, à déterminer son niveau de détail, puis à cumuler le budget mémoire correspondant en utilisant la liste des tailles T décompressées.

Si le budget mémoire n'est pas dépassé alors le meilleur niveau de détail est choisi, sinon un niveau de détail plus bas est choisi.

Ce fonctionnement est illustré sur la figure 20 dans lequel cinq secteurs S0 à S4 sont illustrés. Les métavoxels M hachurés disposés dans les secteurs S3 et S4 devront être chargés avec un niveau de détail bas car la limite de budget est dépassée.

Sur la base des étapes ER1_2 de détermination des métavoxels visibles et ER1_3 de calcul des niveaux de détail, les étapes :

- ER1_4 de demandes de mise à jour de la queue de chargement,
- ER1_5 de demande de mise à jour de la queue de déchargement,
- ER1_6 de chargement effectif et
- ER1_7 de déchargement effectif sont effectuées.

L'objectif de ces étapes est d'éviter une latence trop longue en fragmentant les tâches de chargement et déchargement.

Ainsi, Le thread HDD->RAM effectue une requête de mise à jour. Toutes les t secondes, le thread secondaire RAM->VRAM vérifie les demandes de requêtes de mise à jour des queues de chargement et de déchargement. Si une requête est rencontrée, les métavoxels sont effectivement chargés du HDD vers RAM.

Pour réduire le temps de chargement global, le niveau de détail le plus bas, constitué notamment par les imposteurs I est toujours chargé en mémoire vidéo VRAM.

La mise à jour de la queue de déchargement consiste à décharger des métavoxels de la mémoire vidéo VRAM.

La queue de chargement permet de :

- charger un métavoxel depuis le disque HDD vers la mémoire vive RAM, puis

- dans un second temps, de charger le métavoxel vers la mémoire vidéo VRAM et de le décharger de la mémoire vive RAM.

De plus, les doublons de stockage RAM/VRAM sont évités grâce aux structures de données choisies et en particulier au stockage des métadonnées dans les données géométriques, il n'est pas nécessaire de conserver la géométrie en RAM.

Le calcul des priorités entre les chargements/déchargements a pour objectif de limiter la latence de chargement, de maximiser les gains de mémoires et d'éviter les trous lors de l'affichage.

Pour limiter la latence, il convient de privilégier les transitions faisant gagner un niveau de détail. Pour maximiser les gains en mémoire, il convient de privilégier les déchargements aux chargements et les transitions vers un niveau de détail plus bas. Pour éviter d'obtenir des trous lors de l'affichage, il convient de ne décharger un ancien metavoxel seulement si un metavoxel vient d'être chargé à la place.

Ainsi l'ordre de priorité, à titre d'exemple peut être le suivant :

a- les transitions de tous niveau (n0 à n3) comprenant de la géométrie vers un niveau d'imposteur (n4)

b- les transitions d'un niveau d'imposteur (n4) vers un niveau comprenant de la géométrie avec un niveau de détail important (n0 à n2)

c- les transitions d'un niveau de détail avec géométrie vers un niveau de détail plus important (n2 vers n1 ou n3 vers n2) ;

d- les transitions d'un niveau de détail avec géométrie vers un niveau de détail inférieur avec géométrie (n1 vers n2 ou n2 vers n3) ;

e – les transitions d'un niveau d'imposteur (n4) vers un niveau avec géométrie de faible niveau de détail (n3).

2. Etapes de rendu - Second mode de rendu

Nous allons à présent détailler la réalisation du second mode de rendu. Comme nous l'avons vu précédemment, ce second mode de rendu utilise uniquement le ou les processeurs génériques (CPU), sans utiliser de moyens d'accélération graphique.

L'objectif de ce second mode est de réaliser un rendu le plus rapidement possible en termes de temps de chargement et de fréquence d'affichage en respectant le budget de mémoire vive RAM disponible, en maximisant la qualité de rendu, sans carte vidéo fournissant une accélération 3D, c'est-à-dire sans moyen dédié de rendu de triangles et sans mémoire vidéo (VRAM) disponible.

Les étapes de rendu sont représentées sur la figure 21.

En ce qui concerne l'utilisation des métadonnées, comme nous l'avons vu précédemment, un dictionnaire couples (clé K, valeur V) a été constitué, que les clés ont été injectées dans la charge utile des voxels, et que les valeurs ont été stockées dans un SGDB.

Le traitement d'une requête relative aux métadonnées est réalisé de la façon suivante, similaire à celle du premier mode de rendu.

La clé K est récupérée depuis l'affichage 3D. Ainsi, lors d'une requête de l'utilisateur ER2_0, pour une métadonnée correspondant à un point de l'image, le CPU peut obtenir, lors du rendu, la valeur de la clé stockée dans les données géométriques pour le point considéré.

Le dictionnaire est alors interrogé à l'aide de la clé K dans une étape ER2_9.

En réponse à la requête ER2_9, un retour visuel sur l'affichage 3D peut être effectué dans une étape ER2_5, par exemple sous forme de mise en évidence ou d'occultation d'un objet, ou encore d'affichage des propriétés, de filtrage des données.

En ce qui concerne la réalisation du rendu en lui-même, il apparaît qu'en réponse à une mise à jour de la caméra dans une étape ER2_1 suite à une interaction utilisateur ER2_0, un chargement des données entre le disque et la mémoire vive (RAM) est effectué et un rendu est réalisé.

En ce qui concerne l'ordonnancement de ces tâches, il apparaît qu'au moins deux threads de chargement et de rendu sont entrelacés.

Comme cela est représenté sur la figure 21, 3 niveaux de caches peuvent être utilisés, afin de limiter la consommation mémoire et maximiser la qualité du rendu.

Un premier niveau de cache correspond aux métavoxels et contient les voxels de la grille de niveau 0. Tous les voxels de niveau 0 sont stockés dans ce cache dès le départ du rendu.

Un second niveau de cache correspond aux métavoxels qui contiennent les voxels de la grille de niveau supérieur au niveau 0. Le mode de chargement consiste à charger les voxels fils des voxels G0 jusqu'à saturation de la mémoire. Le mode de chargement dans ce premier cache est le suivant : suite à un calcul des métavoxels2D visibles, les voxels G0 sont chargés jusqu'à saturation de la mémoire, en en chargeant le maximum possible en fonction du point de vue et du déplacement.

Un troisième niveau de cache est un buffer circulaire destiné à charger les voxels correspondant au niveau de profondeur de la grille récursive la plus élevée. Les voxels à charger sont déterminés à l'aide d'un lancer de rayon, lors de la traversée de la grille régulière récursive. Ce troisième cache est un buffer sous forme de buffer circulaire. Lorsqu'un élément du cache est utilisé, son indice de réutilisation est augmenté. Le choix des éléments à remplacer dans le cache correspond à l'élément le moins utilisé dans le cache. Ce troisième cache est utilisé pour un mode de rendu spécifique HD+ qui sera détaillé ultérieurement.

Comme cela est illustré sur la figure 21, suite à la mise à jour de la caméra, une étape ER_2_6 de détermination du mode de rendu est réalisée, afin de déterminer un mode de rendu permettant d'afficher le plus rapidement avec la meilleure qualité possible.

En particulier, 4 modes d'affichage sont définis.

Dans un premier mode à basse définition ou SD, des voxels de la grille de niveau 0 et éventuellement de niveau supérieur sont affichés. Des requêtes de chargement des métavoxels de la grille de niveau supérieur à 0 sont lancées et les métavoxels correspondants sont chargés au fur et à mesure que la caméra se déplace. Pour éviter la latence liée au chargement des métavoxels de niveau supérieur à 0, le rendu est effectué avec les métavoxels de niveau supérieur à 0 qui sont disponibles suite à leur chargement et le reste des métavoxels visibles conserve le niveau 0.

La définition de la fenêtre de rendu est diminuée, par exemple en étant divisée par deux. Ce mode est activé lors d'un déplacement de la caméra;

Dans un second mode à moyenne définition ou MD activé dès que le mouvement de la position de la caméra est arrêté, un affichage à la définition de la fenêtre de rendu est réalisé. Les métavoxels de niveau de grille supérieur à 0 chargés pendant le déplacement dans le mode SD sont pris en compte. Le reste des métavoxels visible conserve le niveau G0. Un recalcul de l'image à la dimension de la fenêtre est réalisé.

Si la position de la caméra reste inchangée, lorsque l'ensemble des requêtes de chargement est réalisée, un nouveau rendu est réalisé selon un troisième mode HD avec les métavoxels de niveau 0 et de niveau supérieur chargés selon des niveaux de priorité jusqu'à saturation du second cache.

Ces trois premiers modes utilisent le premier et le second cache décrits précédemment.

Sur requête de l'utilisateur, un quatrième type de rendu HD+ peut être réalisé, à la définition de la fenêtre de rendu. Dans ce cas, tous les métavoxels de niveau de détail le plus élevé, visibles depuis la caméra sont chargés. Afin de pouvoir tous les considérer, ils sont chargés de manière circulaire dans le troisième cache.

La figure 22 illustre les relations et les activations des différents modes de rendu. A partir d'un état « inactif », si un événement « en mvt » correspondant à un mouvement de caméra est détecté, une transition vers un état de calcul d'une image SD « calcul image SD » est déclenchée. A partir de cet état, une transition vers un état « inactif en mouvement » est déclenchée lorsque l'image SD est prête. Une transition en retour vers l'état « calcul image SD » est déclenchée si les paramètres de caméra sont modifiés « MAJ Caméra ». A partir des états « calcul image SD » ou « Inactif en mvt », une transition est déclenchée vers un état de calcul d'une image MD lors de la détection d'une fin de mouvement de caméra « fin de mvt ».A partir de cet état un transistion vers un état intemédiaire lorsque le calcul de l'image MD est achevé. Dans cet état intermédiaire, une transition vers l'état « Inactif » est déclenchée si une requête de chargement « RC (requête de chargement) en cours » est en cours et une transition vers un état de calcul d'une image en mode HD « Calcul image HD » si aucune requête de chargement n'est en cours « Pas de RC en cours ».

A partir de l'état « Inactif », une transition vers l'état « Calcul image HD » est générée si l'ensemble des requêtes de chargement ont été traitées « Aucune RC en cours ».

Sur requête d'un utilisateur, un calcul d'une image en mode HD+ peut être déclenché dans un état « calcul image HD+ », sous réserve qu'une image HD ait déjà été calculée.

A partir des états « calcul image MD », « calcul image HD » ou « calcul image HD+ », une transition est générée vers un état d'arrêt des requêtes de chargement et de vidage du cache « Arrêt et effacement du Cache » sur une détection d'une reprise du mouvement « En mvt ». Une fois l'opération de vidange du cache réalisée, une transition vers l'état « Calcul image SD » est déclenchée.

De façon optionnelle, un état de calcul sans affichage d'une image SD peut être prévu, afin d'optimiser le calcul des objets visibles par un test d'occlusion (voxels cachés par d'autres voxels). Un rendu SD permet en effet de construire l'ensemble des métavoxels V1 visibles depuis la caméra, seuls ces voxels étant ensuite chargés lors du calcul d'une image MD ou HD.

En ce qui concerne l'étape ER2_4 de rendu des voxels, ce rendu peut notamment être réalisé par la méthode dite du 3D-DDA, illustrée sur la figure 23.

Chaque élément de la grille est examiné le long du rayon. La traversée de la grille est récursive. Nous prenons ci-dessous un exemple en référence à la figure 21 et en se limitant à deux niveau de grille. Bien entendu, des niveaux de grille supplémentaires peuvent être utilisés.

Tant que le rayon n'a pas rencontré d'élément au niveau 0 de la grille, le curseur avance le long du rayon. Lorsqu'un élément plein au niveau 0 est rencontré par le rayon, comme par exemple le voxel V0 sur la figure 21, le rayon parcours les éléments de niveau 1 contenus dans le métavoxel v0. Tant que le rayon n'a pas rencontré d'élément 1 de la grille dans le métavoxel, le curseur avance le long du rayon. Lorsque un élément plein est rencontré, comme par exemple le voxel v3 de la figure 21: la traversée s'arrête, l'éclairage est calculé, le cache de voxels est mis à jour.

V. Relevés de performance

La figure 24 reprend des données de relevé de performances sur une plateforme de tests constituée par un ordinateur grand public, dont l'unité

centrale comprend un processeur 4 cœurs (Intel Core2Quad Q6600, 2.40 Ghz), une mémoire vive (RAM) de 2 Go, une carte graphique: NVidia GeForce 8800 GTX avec une mémoire vidéo (VRAM) de 768 Mo dont la capacité maximale effective est de 30 millions de triangles @ 10fps.

Les relevés de performance ont été effectués sur la base du rendu avec GPU de deux modèles:

- un premier modèle Md1 comprenant 35Go de données, 30 milliards de triangles, 600000 objets et 30 groupes; et
- un second modèle Md2 comprenant 10Go de données, 40 millions de triangles, 20000 objets et 7 groupes.

Selon des variantes de l'invention, dans la phase de préparation des données, d'autres mécanismes de simplification du maillage peuvent être utilisés. Il est également possible de considérer d'autres sources de données à l'aide de connecteurs.

Dans la phase de rendu des données, d'autres mécanismes peuvent être également utilisés pour afficher des données en très basse qualité (imposteurs/voxels...), des voxels rendus par CPU/GPU, et/ou utilisant un rendu hybride mettant en œuvre le CPU/GPU (CPU/CPU ou GPU/GPU)

Il est possible d'utiliser d'autres moteurs de rendu (GPU/CPU) à l'aide de connecteurs.

REVENDICATIONS

1. Procédé de rendu d'une scène virtuelle en trois dimensions comprenant:

une première phase de préparation des données comprenant au moins une étape de regroupement des données de la scène dans une structure de données (OD) correspondant à une division spatiale de la scène en une grille régulière de façon à former des métavoxels correspondant à la séparation d'un modèle de scène en un ensemble de sous-modèles suivant le découpage de ladite grille régulière , et

une seconde phase de rendu temps-réel utilisant la structure de donnée géométrique (OD) pour le rendu d'au moins une image,

la première phase de préparation des données comprenant au moins une étape d'injection de clés (K) relatives à des valeurs de métadonnées (V) dans ladite structure de donnée (OD) et la constitution d'une base de donnée comprenant les valeurs (V) des métadonnées correspondants aux clés (K) stockées dans la structure de donnée géométriques (OD), de façon à permettre, lors d'une interaction utilisateur dans la seconde phase de rendu, la mise à disposition d'une valeur de métadonnée extraite de la base en lien avec un élément d'image considéré correspondant au rendu d'un métavoxel.

2. Procédé selon la revendication 1, dans lequel chaque métavoxel comprend un ensemble de triangles, les triangles intersectés des objets sont stockés de manière contiguë en conservant une métadonnée (M) correspondant à un identifiant de l'objet pour les différencier.

3. Procédé selon l'une des revendications précédentes, dans lequel les données géométriques de la scène sont séparées en groupes, chaque groupe étant traité indépendamment lors de l'étape de formation des métavoxels.

4. Procédé selon l'une des revendications précédentes, dans lequel lors de la phase de préparation, les groupes d'objets sont concaténés ou fusionnés, les données géométriques étant stockées de manière contigüe pour chaque métavoxel en conservant une métadonnée sous forme d'un identifiant de groupes et d'objets.

5. Procédé selon la revendication 4, dans lequel les identifiants de groupe et d'objets sont concaténés.

6. Procédé selon l'une des revendications précédentes, dans lequel des moyens d'accélération graphique matériels (GPU) sont utilisés pour procéder à un rendu en associant une clé (K) à un code couleur dans un tampon de rendu permettant d'identifier les clés (K) dans ce tampon.

7. Procédé selon l'une des revendications précédentes, dans lequel une grille régulière récursive est utilisée qui comporte au moins deux niveaux de définitions de grille régulière, les clés (K) d'indexation des métadonnées étant fractionnées et stockées de façon réparties sur les différents niveaux de grille.

8. Procédé selon la revendication 7, dans lequel des moyens d'accélération graphique matériels (CPU seulement) ne sont pas utilisés, la clé (K) est reconstruite lors du rendu du voxel en réunissant les parties de la clé associées aux charges utiles associées aux voxels des différents niveaux de définition de la grille lors du rendu, en interrogeant la charge utile du voxel traversé par le rayon.

9. Procédé selon la revendication 8, dans lequel les objets de la scène 3D sont rassemblés en groupes, et lors de la sélection d'un élément de la scène, il est possible à partir du métavoxel de plus faible niveau d'obtenir une information sur les groupes d'objets présents dans ledit métavoxel, puis de croiser cette information avec les identifiants des objets stockées au niveau du métavoxel de niveau supérieur pour limiter la sélection d'objets possibles.

10. Système destiné à la mise en œuvre d'un procédé selon l'une des revendications précédentes, comprenant :

- au moins un ordinateur permettant l'exécution d'un module de préparation mettant en œuvre les étapes de préparation des données et d'un module de rendu; et
- un système de gestion de base de données stockant les valeurs de métadonnées.

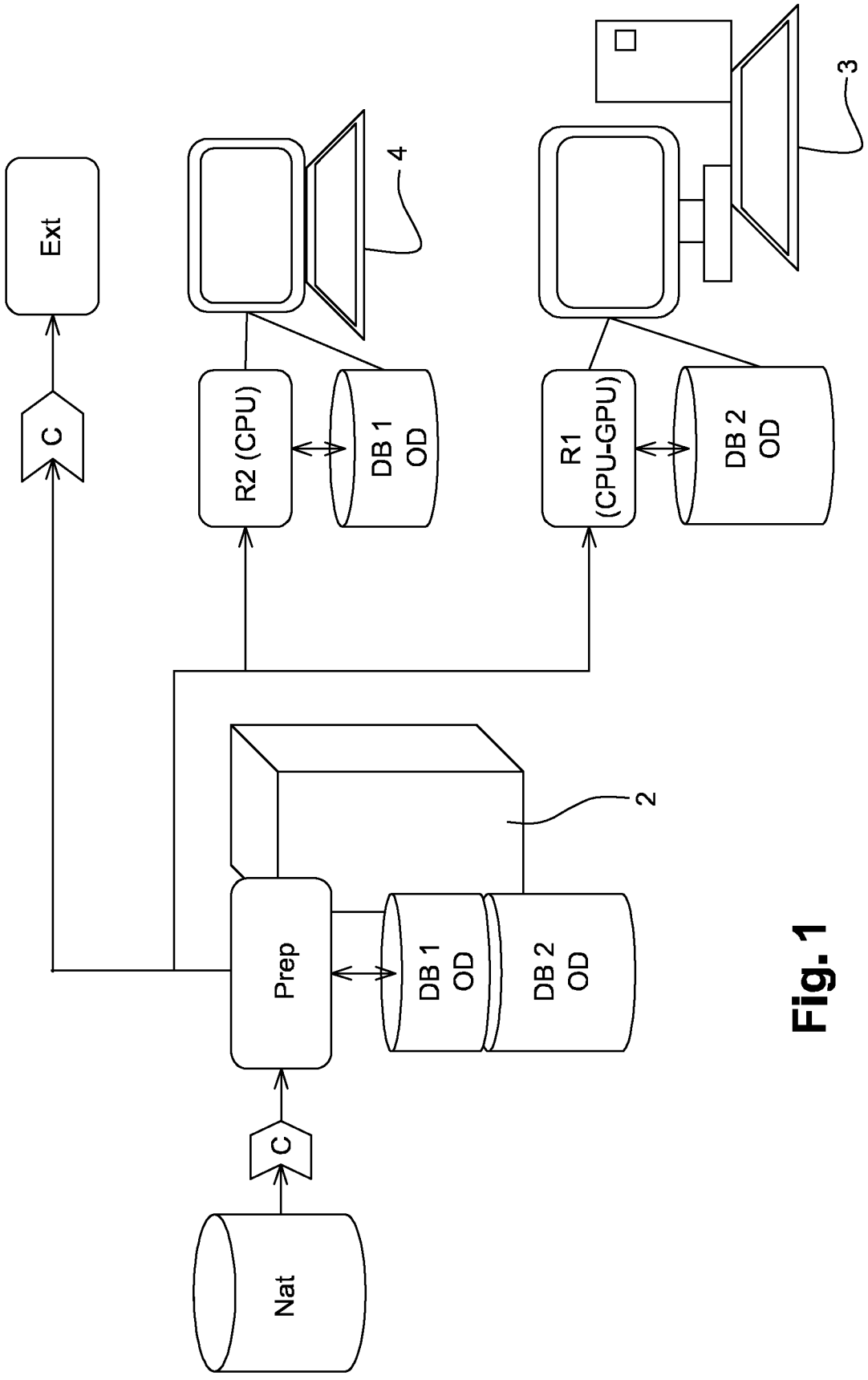


Fig. 1

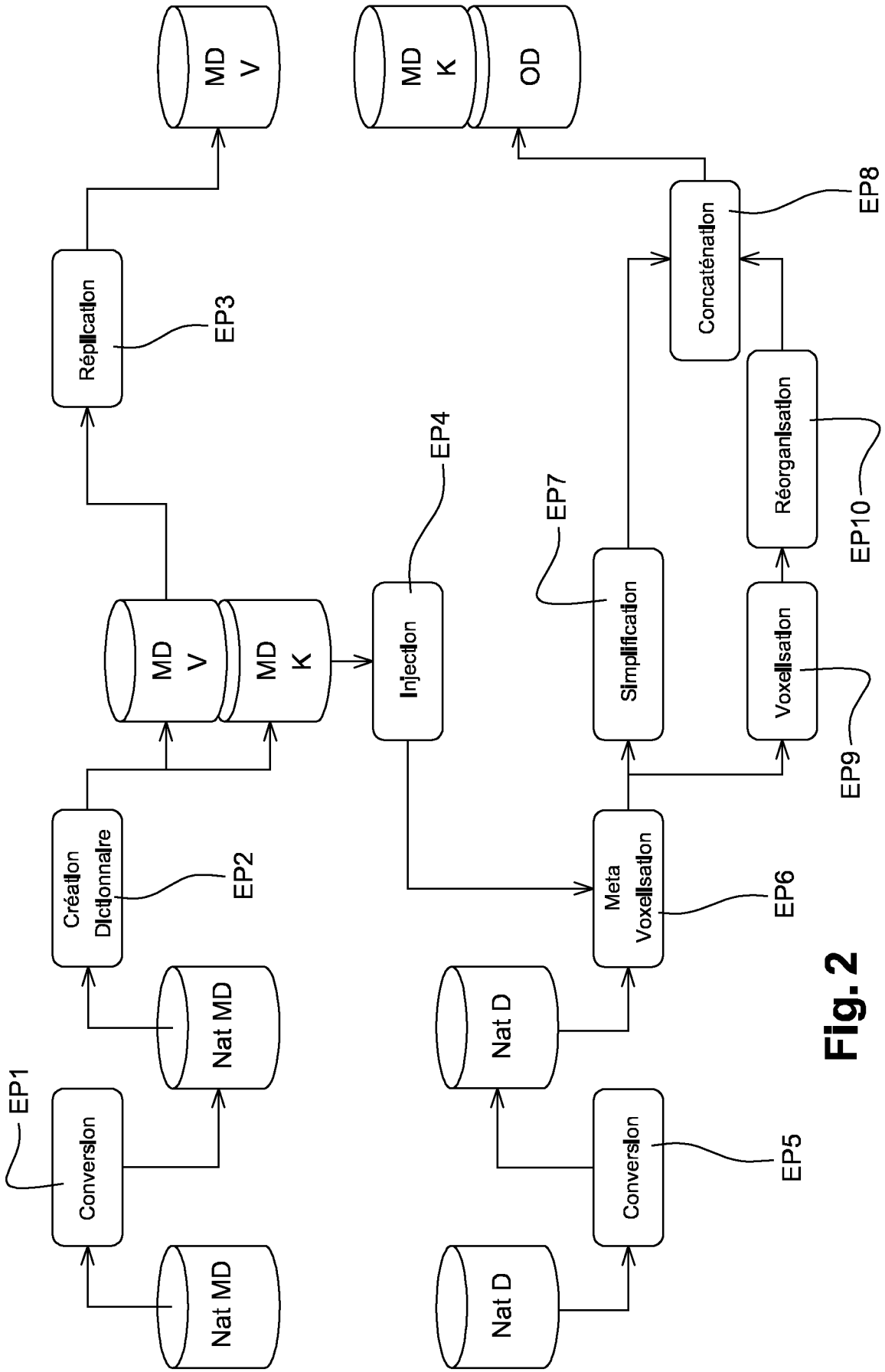

Nat MD
Nat MD
Nat D
Nat D
Conversion
Conversion
Création Dictionnaire
Meta Voxellisation
Réplication
Injection
Simplification
Voxellisation
Concaténation
Réorganisation
MD V
MD K
MD V
MD K
OD
EP1
EP2
EP3
EP4
EP5
EP6
EP7
EP8
EP9
EP10

Fig. 2

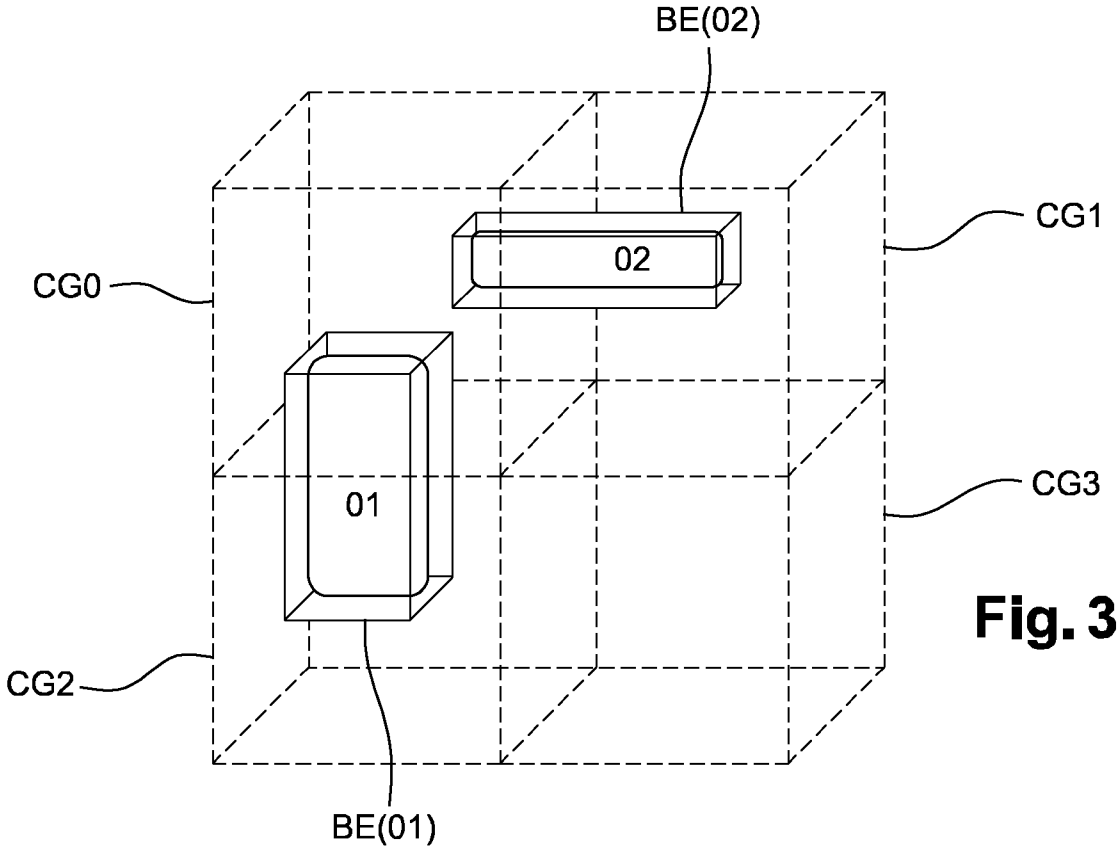


Fig. 3

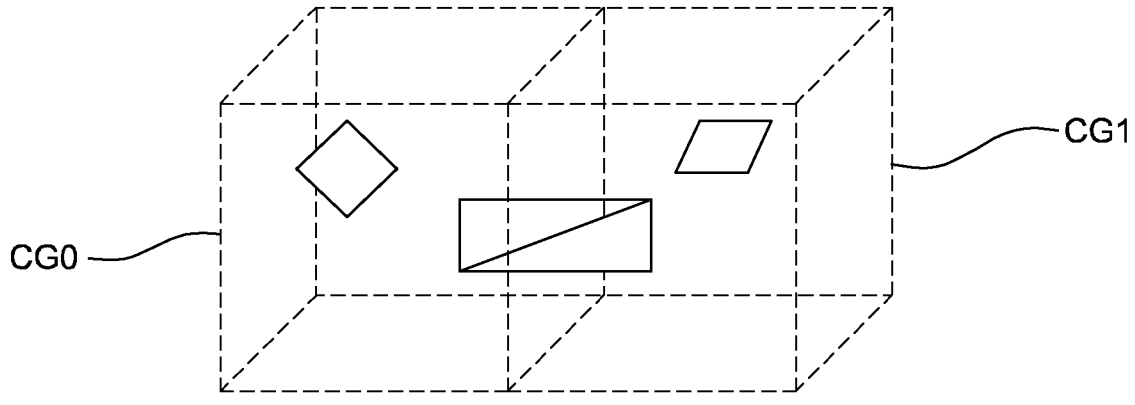
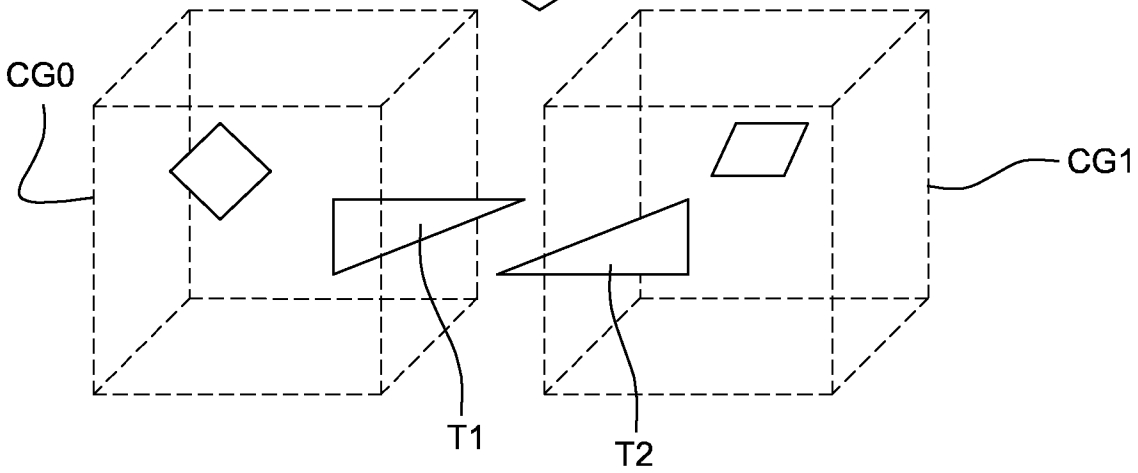


Fig. 4



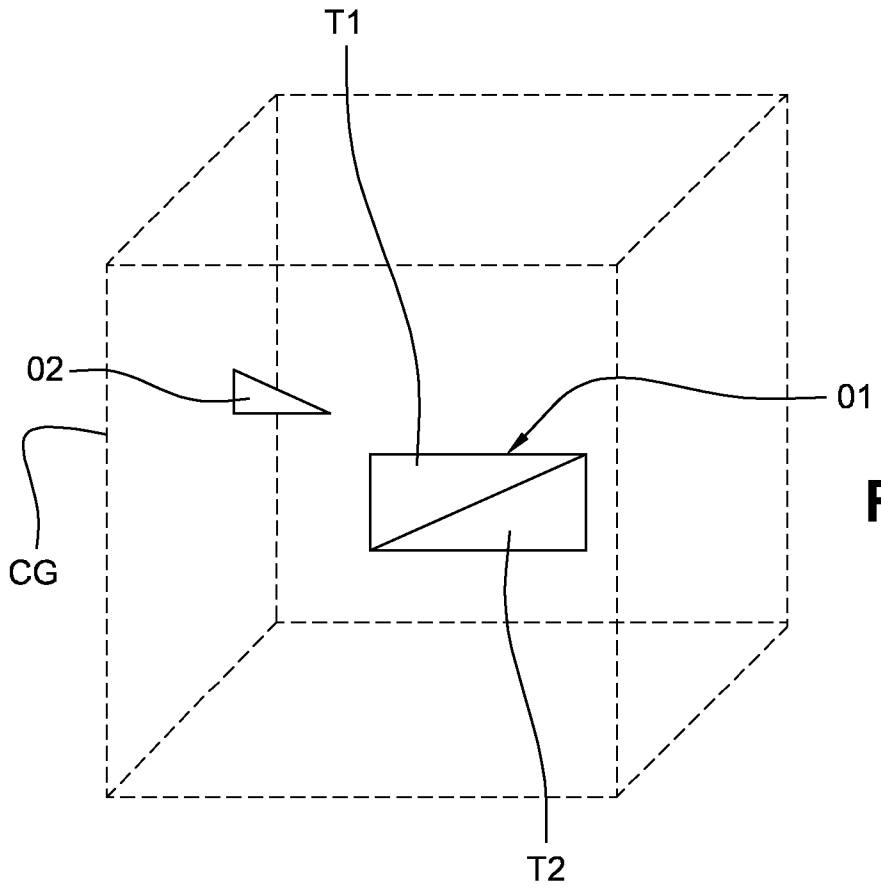


Fig. 5

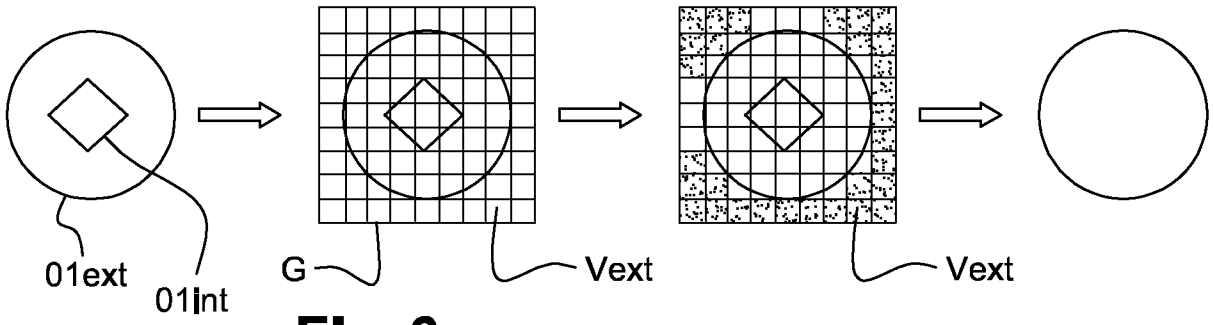


Fig. 6

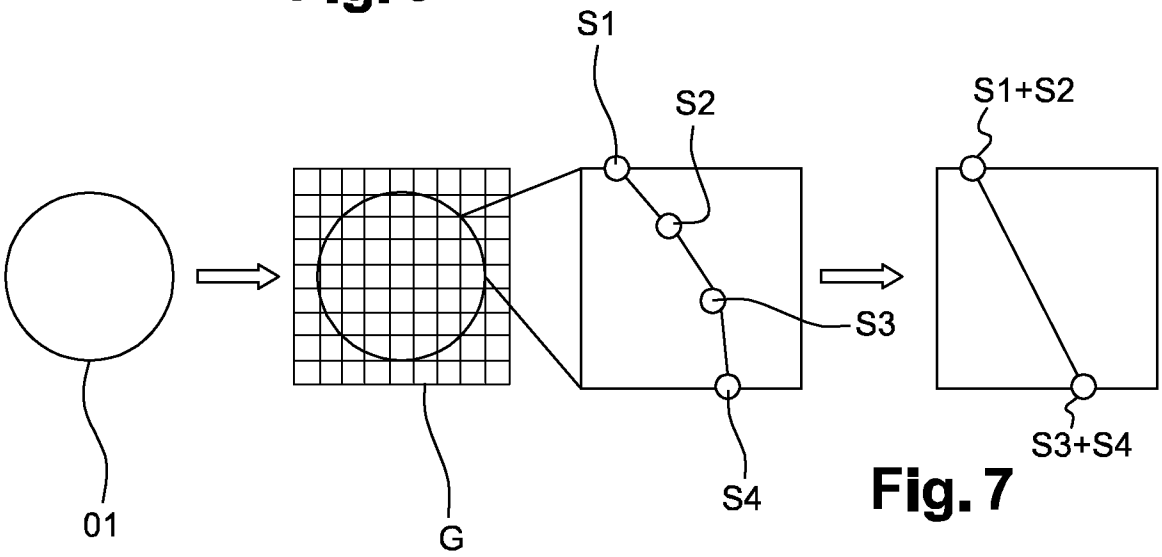


Fig. 7

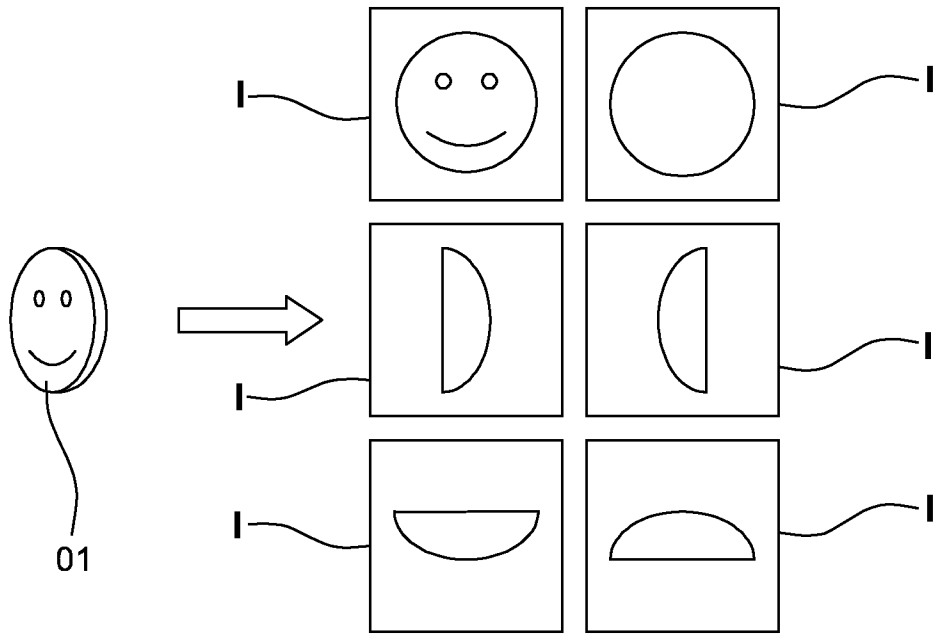


Fig. 8

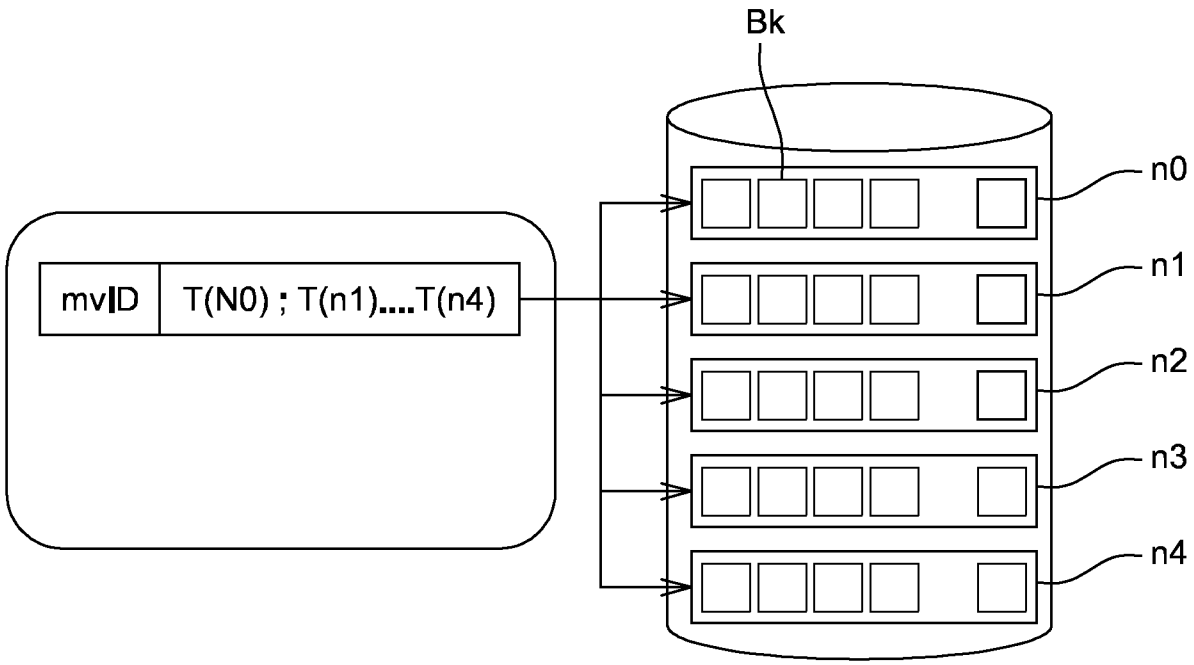


Fig. 9

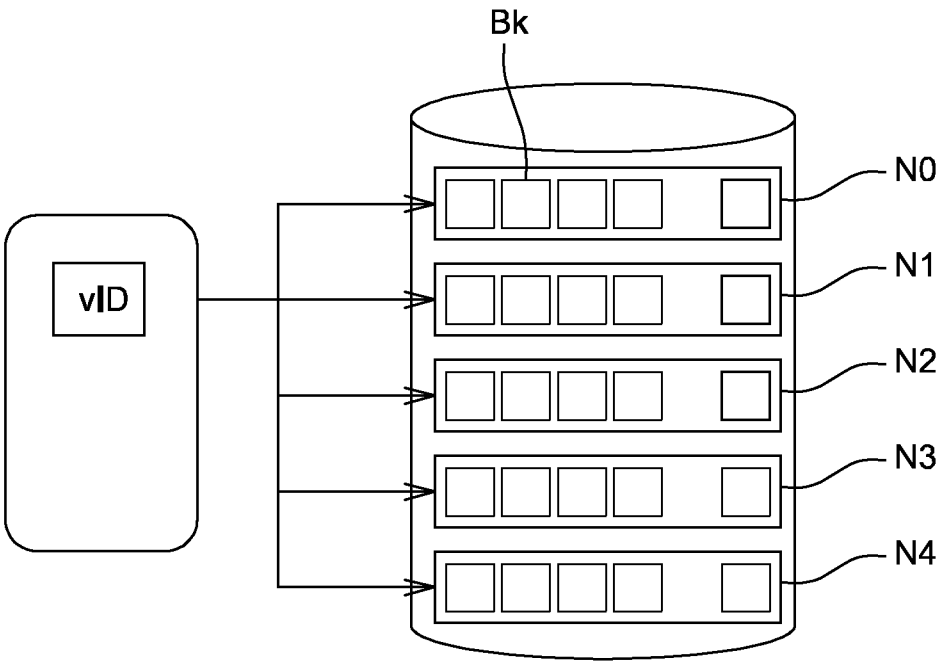

Bk
vID
N0
N1
N2
N3
N4

Fig. 10

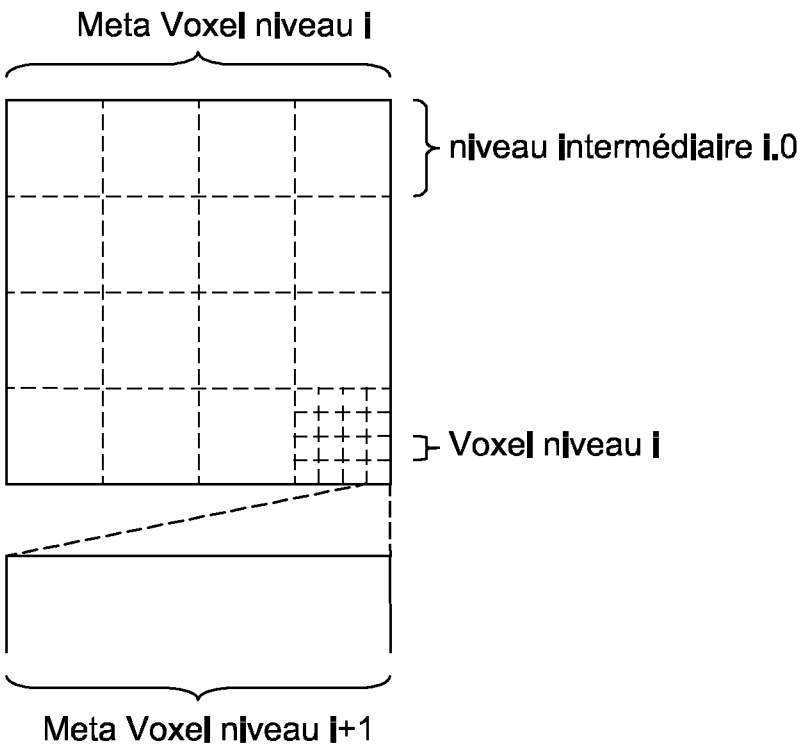

Meta Voxel niveau i
niveau intermédiaire i.0
Voxel niveau i
Meta Voxel niveau i+1

Fig. 11

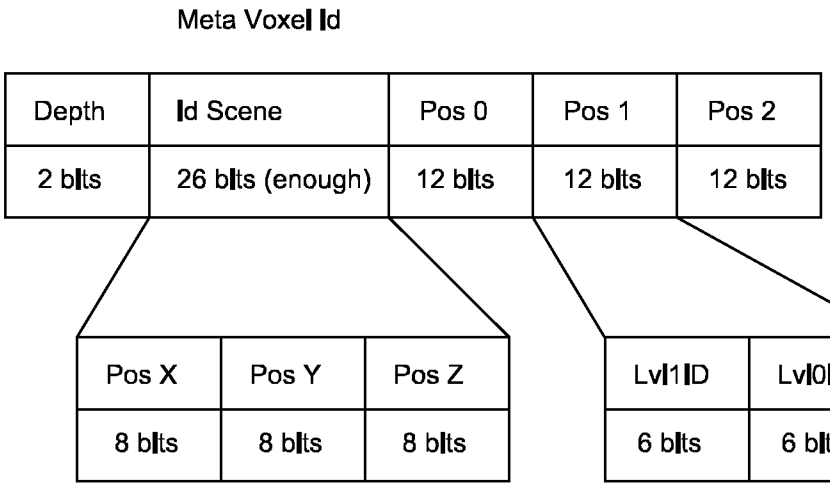


Fig. 12

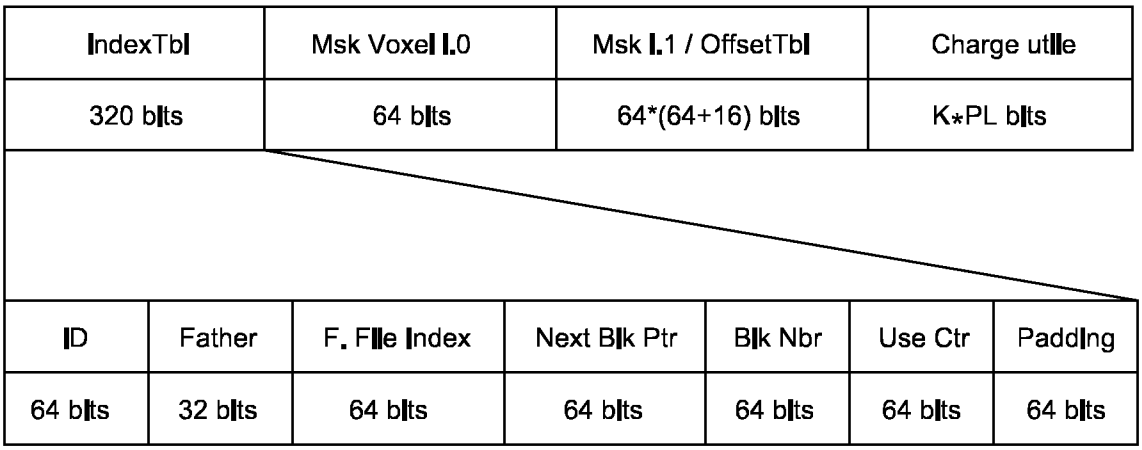


Fig. 13a

ligne	Mask i.0	Mask i.1	Offset
0	1	[0,1,0,1]	0
1	0	0	0
2	1	[1,0,1,1]	2*PISize
3	1	[0,0,1,1]	(2+3)*PISize

Fig. 13b

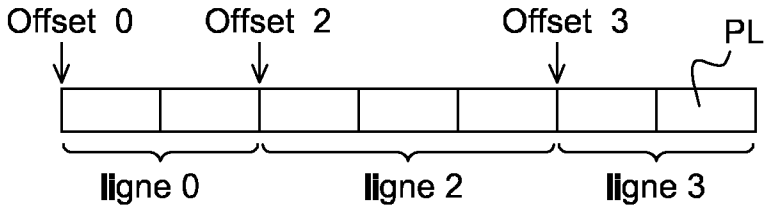


Fig. 13c

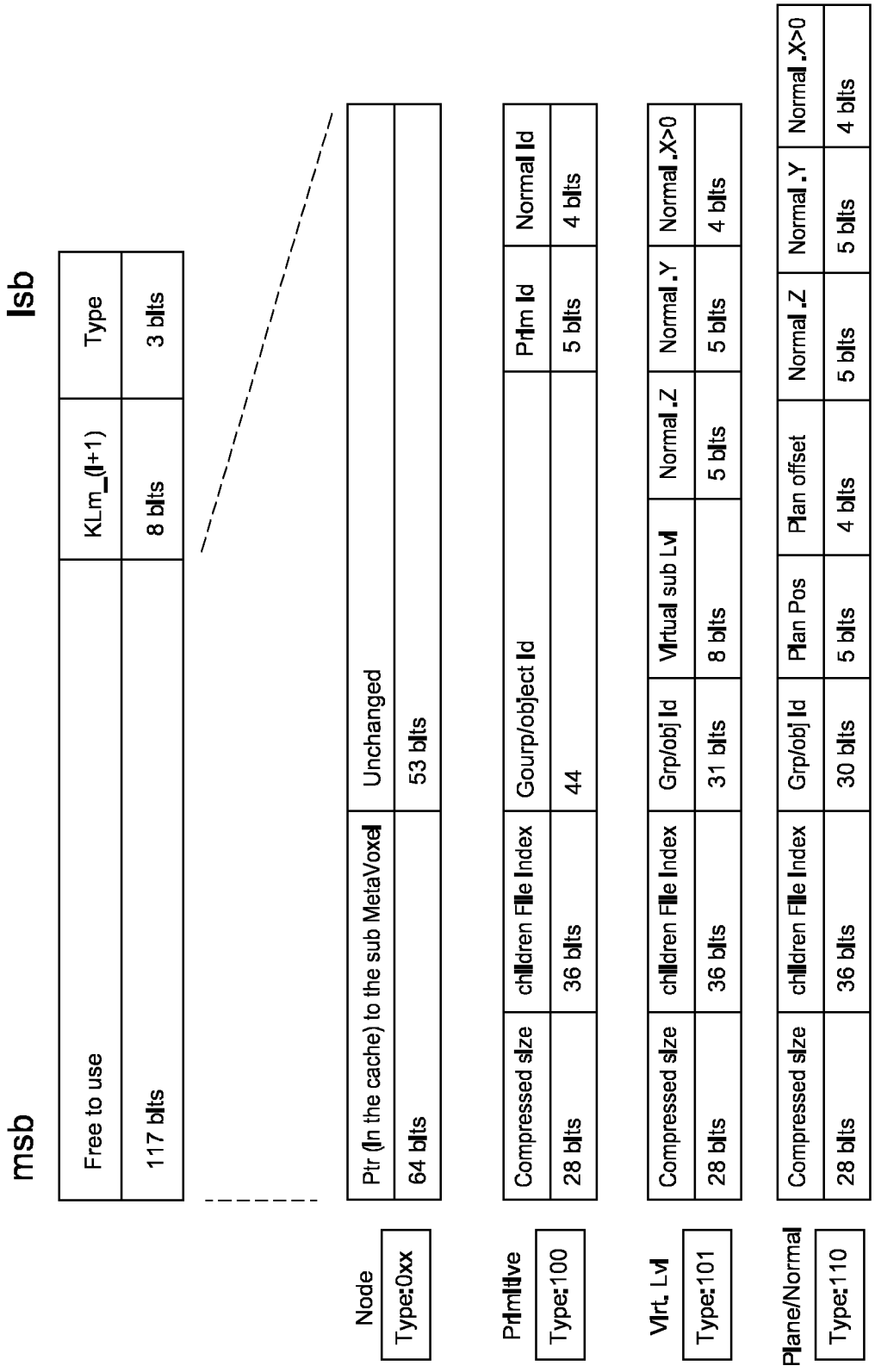
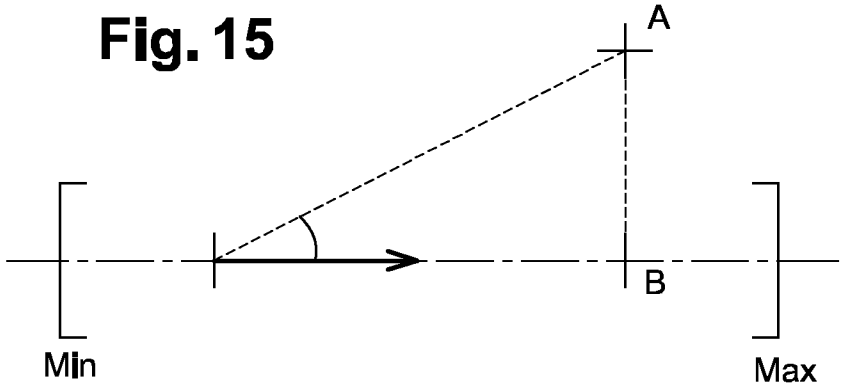

msb
lsb
Free to use
KLm_(l+1)
Type
117 bits
8 bits
3 bits
Node
Type:0xx
Ptr (In the cache) to the sub MetaVoxel
Unchanged
64 bits
53 bits
Primitive
Type:100
Compressed size
children File Index
Gourp/object Id
Prim Id
Normal Id
28 bits
36 bits
44
5 bits
4 bits
Virt. Lvl
Type:101
Compressed size
children File Index
Grp/obj Id
Virtual sub Lvl
Normal .Z
Normal .Y
Normal .X>0
28 bits
36 bits
31 bits
8 bits
5 bits
5 bits
4 bits
Plane/Normal
Type:110
Compressed size
children File Index
Grp/obj Id
Plan Pos
Plan offset
Normal .Z
Normal .Y
Normal .X>0
28 bits
36 bits
30 bits
5 bits
4 bits
5 bits
5 bits
4 bits

Fig. 14

Fig. 15



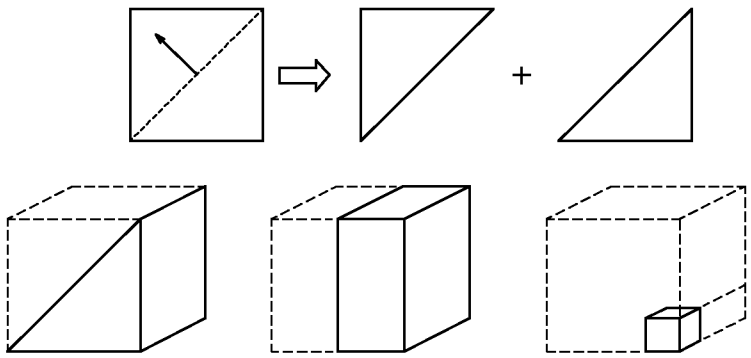


Fig. 16

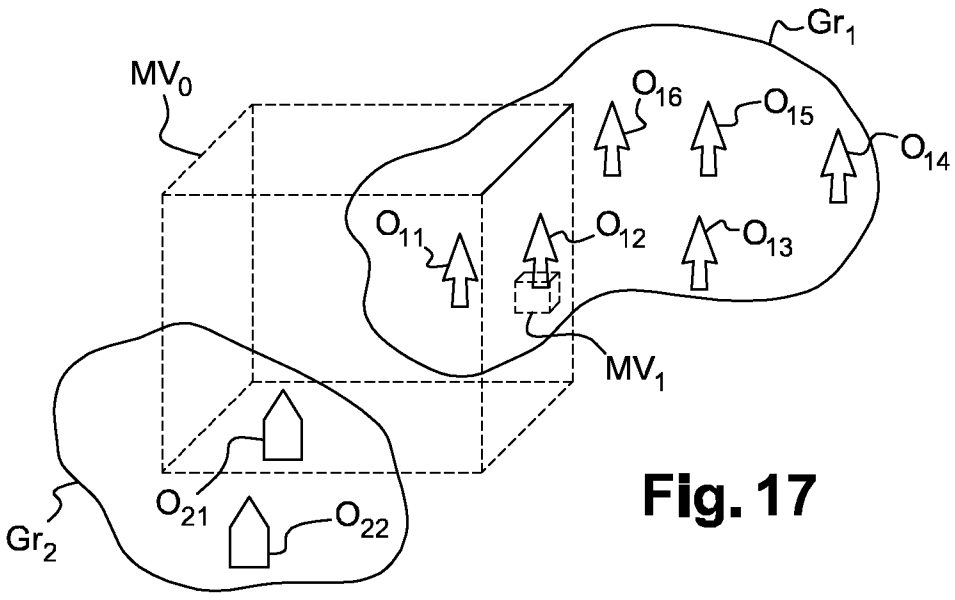


Fig. 17

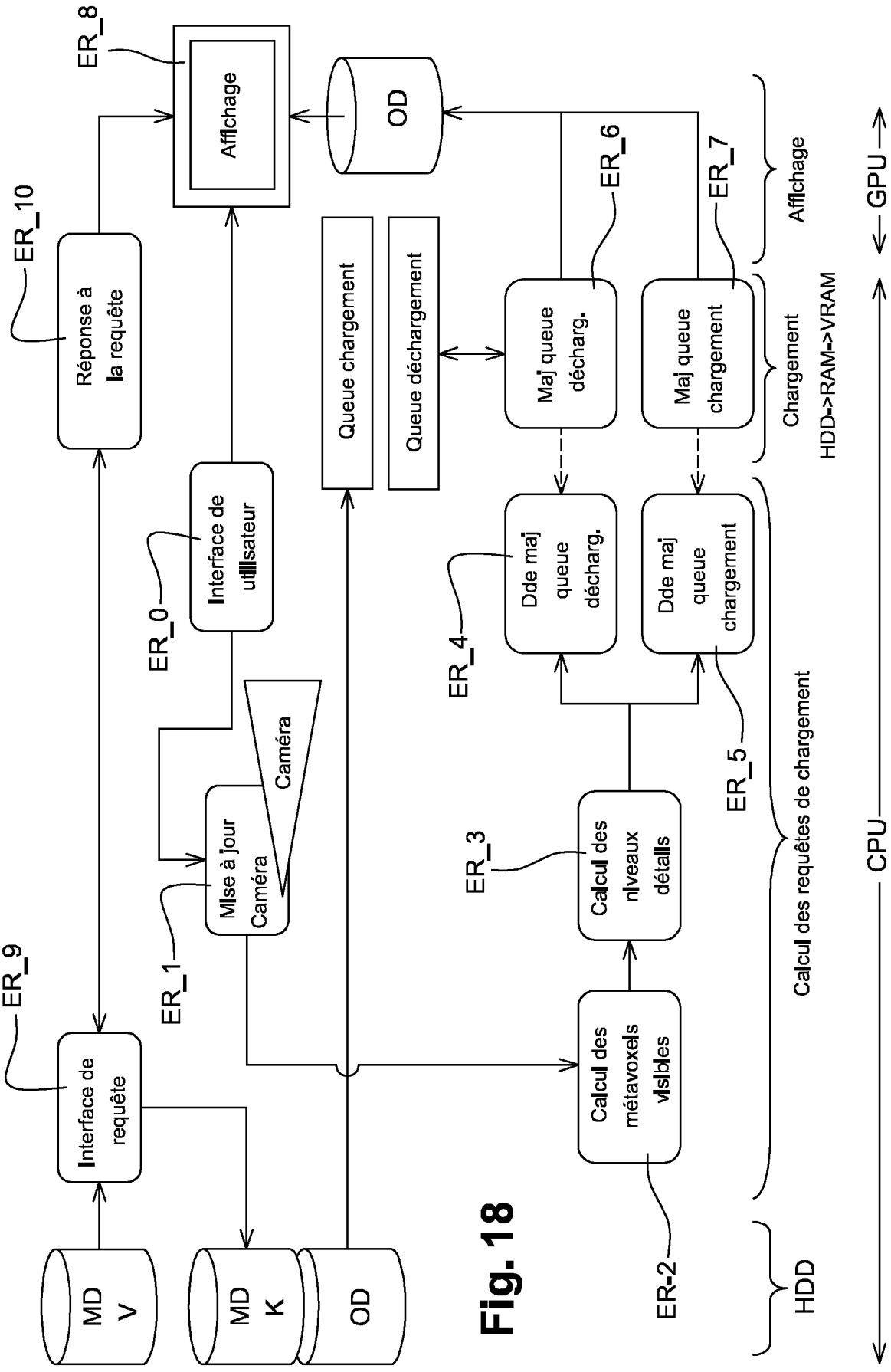

ER_8
Affichage
OD
ER_10
Réponse à la requête
Queue chargement
Queue déchargement
ER_6
Maj queue décharg.
ER_7
Maj queue chargement
Affichage
Chargement HDD->RAM->VRAM
ER_0
Interface de utilisateur
ER_4
Dde maj queue décharg.
Dde maj queue chargement
Caméra
Mise à jour Caméra
ER_1
ER_3
Calcul des niveaux détails
ER_5
ER_9
Interface de requête
Calcul des métavoxels visibles
Calcul des requêtes de chargement
MD V
MD K
OD
ER-2
HDD
CPU
GPU

Fig. 18

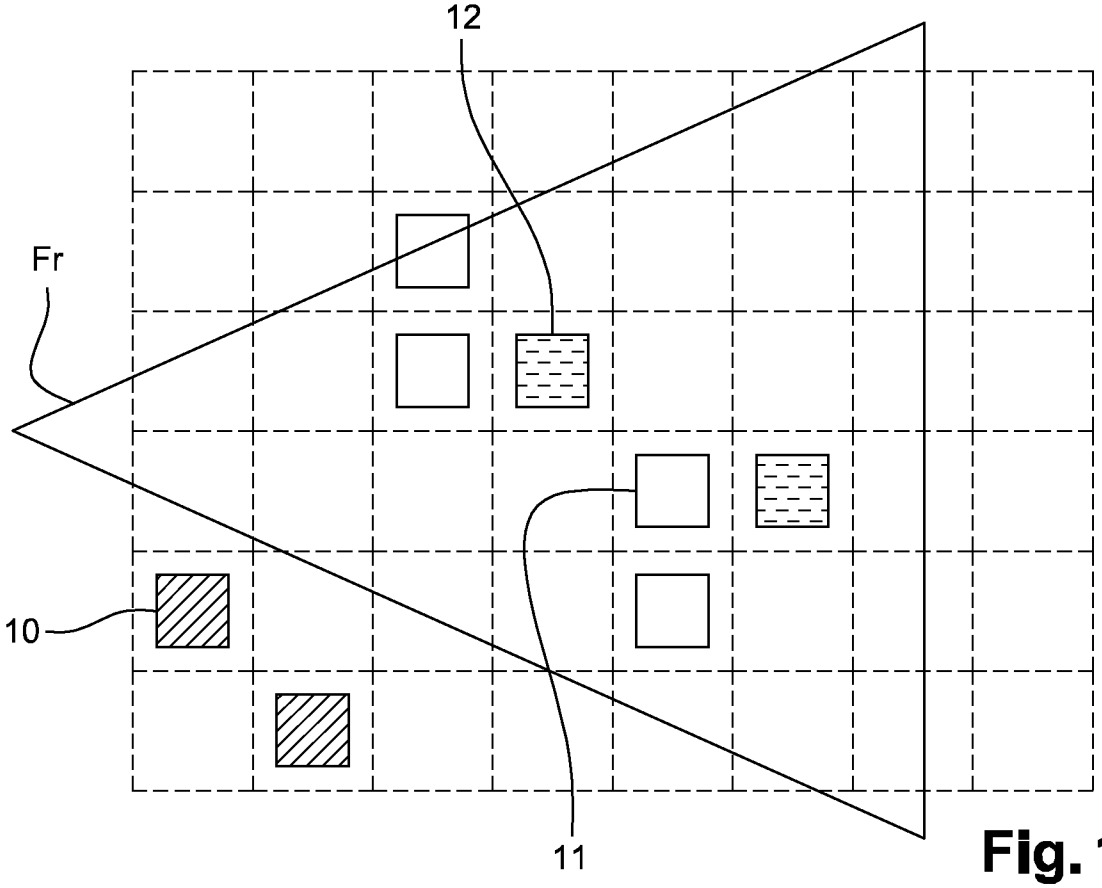


Fig. 19

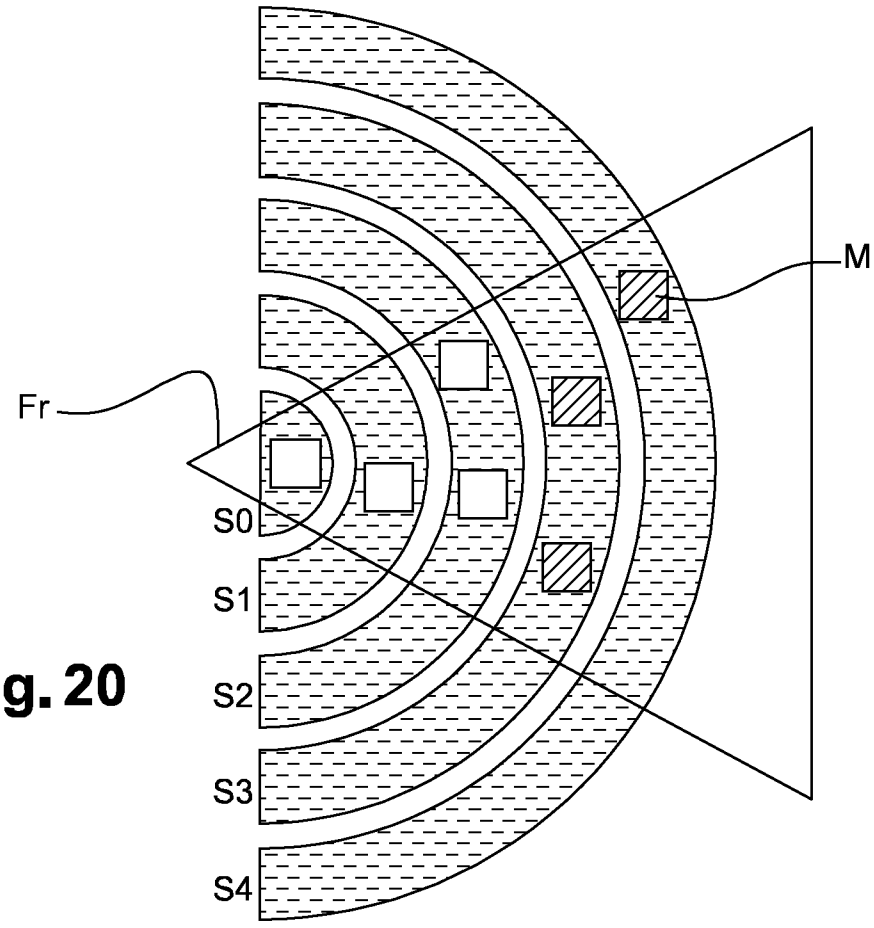


Fig. 20

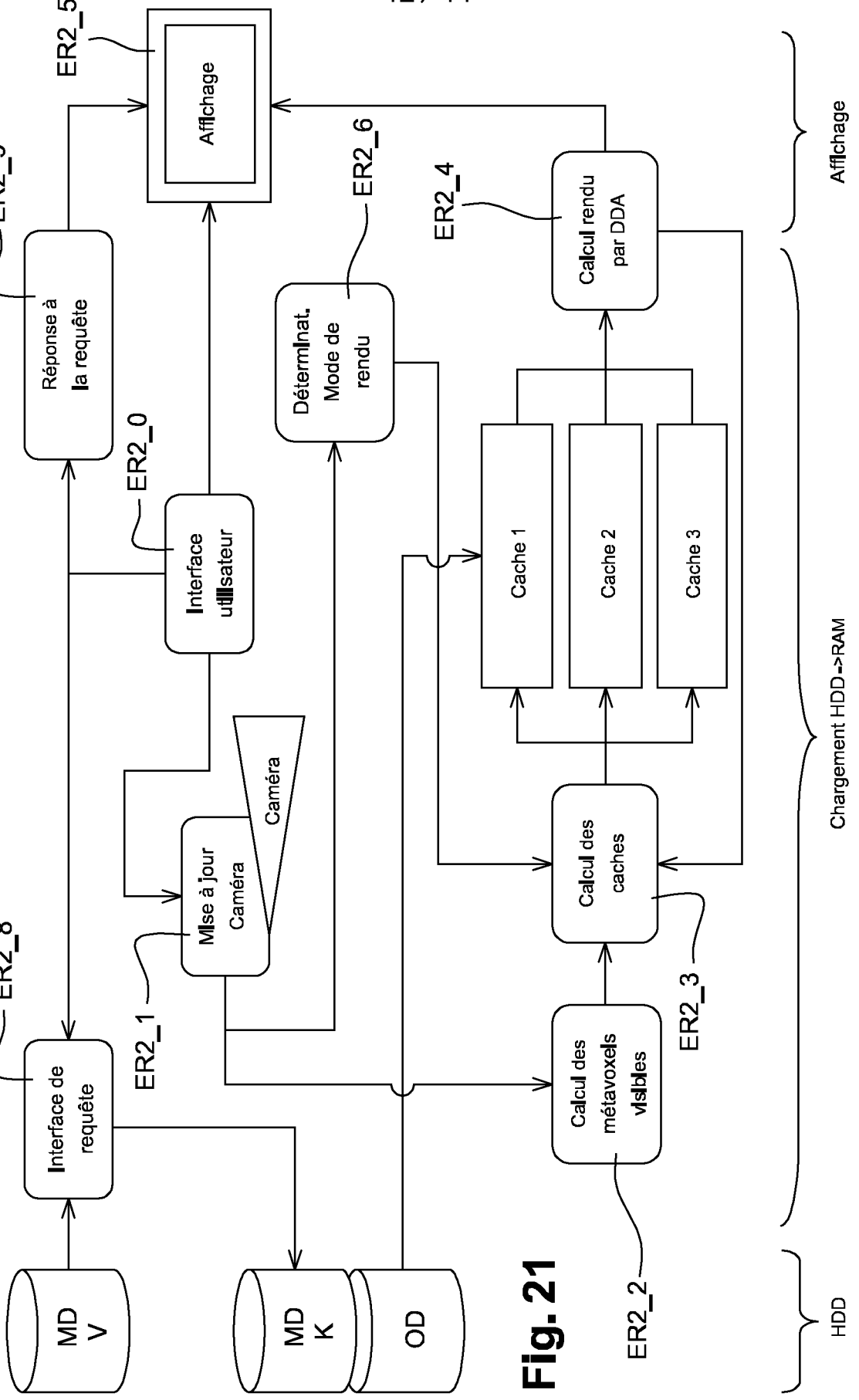

ER2_5
Affichage
ER2_9
Réponse à la requête
ER2_6
Détermination Mode de rendu
ER2_4
Calcul rendu par DDA
Affichage
ER2_0
Interface utilisateur
Cache 1
Cache 2
Cache 3
Caméra
Mise à jour Caméra
ER2_1
ER2_8
Interface de requête
Calcul des caches
Calcul des métavoxels visibles
ER2_3
Chargement HDD->RAM
MD V
MD K
OD
ER2_2
HDD

Fig. 21

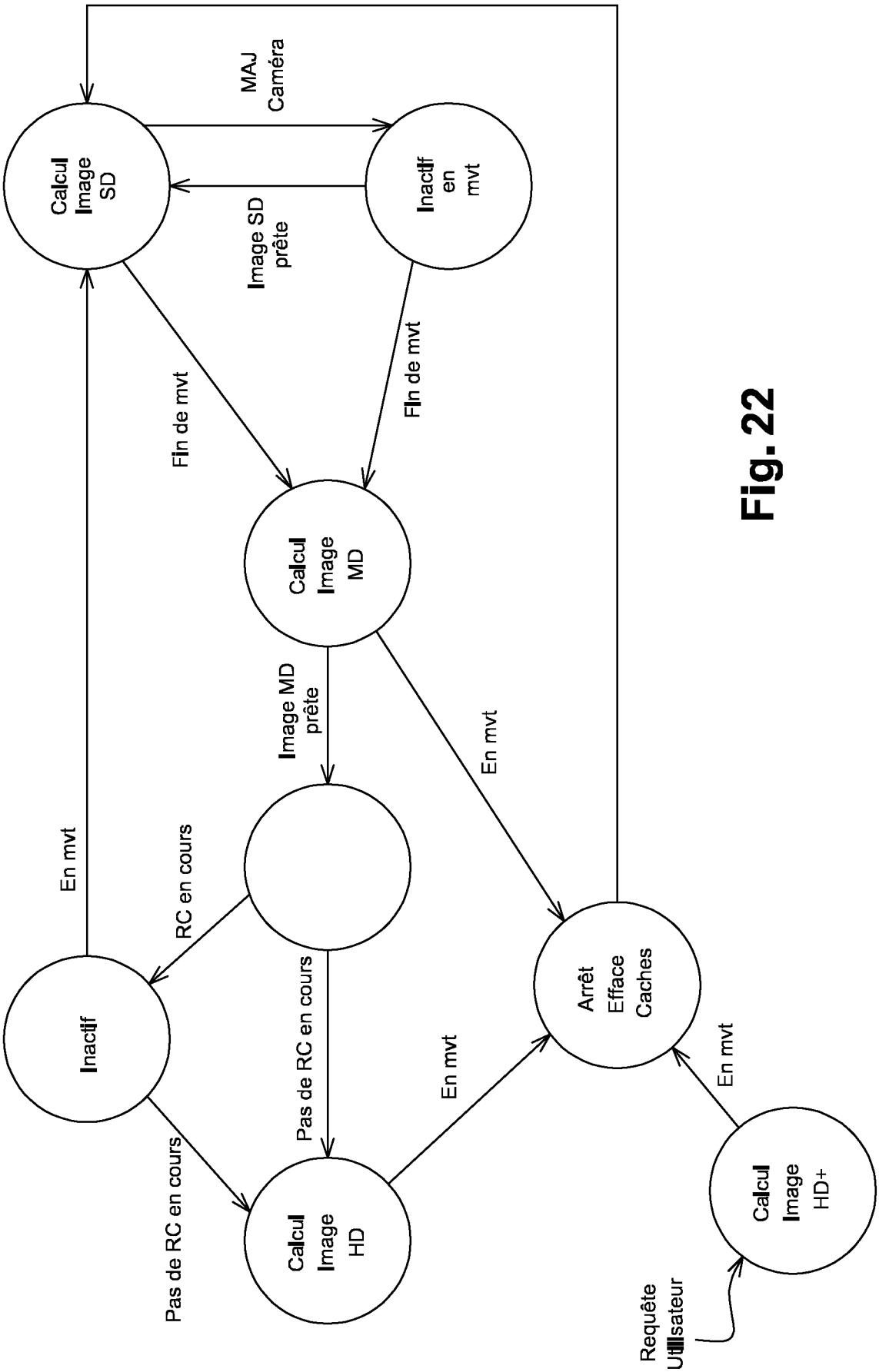

Calcul Image SD
MAJ Caméra
Inactif en mvt
Image SD prête
Fin de mvt
Fin de mvt
Calcul Image MD
Image MD prête
En mvt
En mvt
RC en cours
Inactif
Pas de RC en cours
Pas de RC en cours
Calcul Image HD
En mvt
Arrêt Efface Caches
En mvt
Calcul Image HD+
Requête Utilisateur

Fig. 22

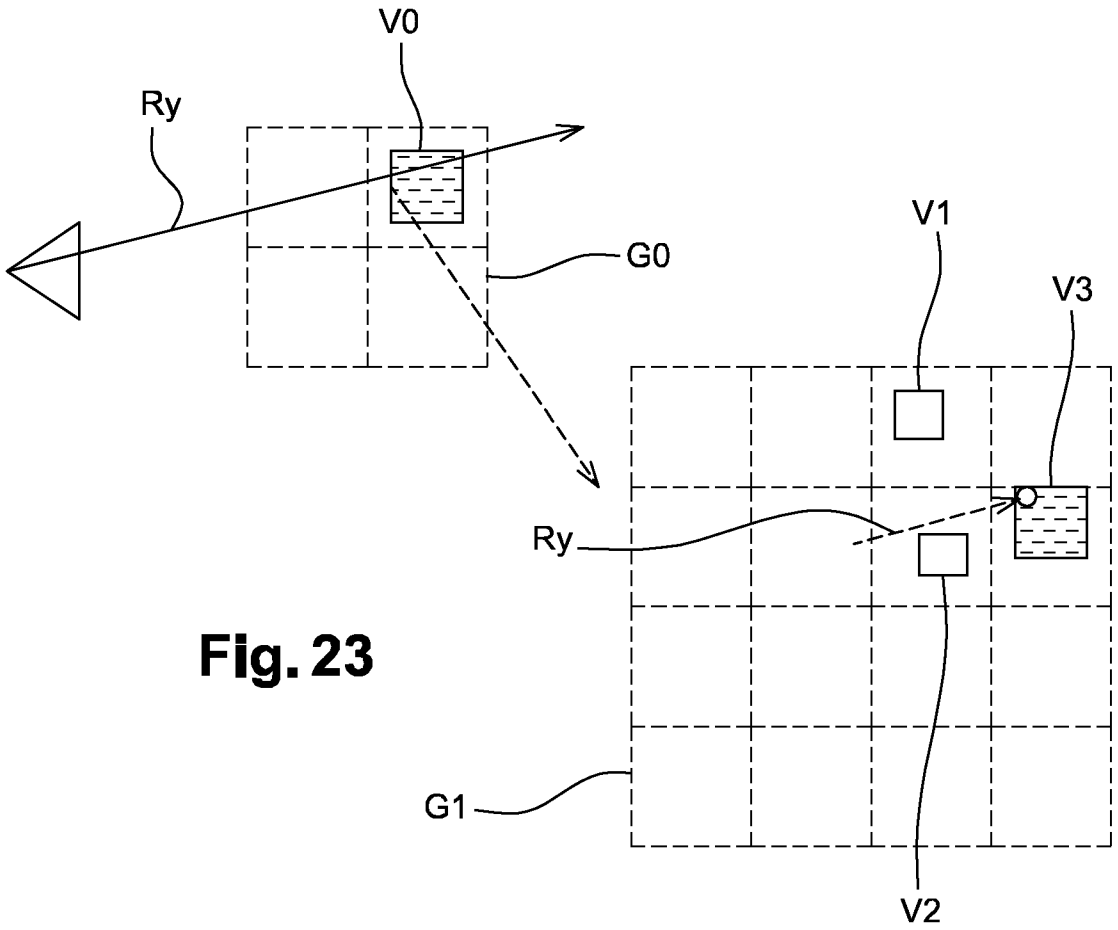


Fig. 23

Modèle	Durée Préparation	Taille BD R1	Taille BD R2	Taille MD	Images p.s.	Occupation Mémoire
Md1	11 h	2500 Voxels 15 Go	2 Go	1 Go	10	<2 Go
Md2	12 h	300 Voxels 4 Go	400 Mo	100 Mo	12	<2 Go

Fig. 24

INTERNATIONAL SEARCH REPORT

International application No
PCT/FR2012/050784

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06T15/00
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06T

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	VARADHAN G ET AL: "Out-of-core rendering of massive geometric environments", VIS 2002. IEEE VISUALIZATION 2002. PROCEEDINGS. BOSTON, MA, OCT. 27 - NOV. 1, 2002; [ANNUAL IEEE CONFERENCE ON VISUALIZATION], NEW YORK, NY : IEEE, US, 1 November 2002 (2002-11-01), pages 69-76, XP031212949, ISBN: 978-0-7803-7498-0 Sections 1 - 4.3, 6 et sous-sections -----	1-10
A	VIVANLOC V: "Rendu distribué sur grappe de CPU/GPU et effects d'éclairage global", THÈSE L'UNIVERSITÉ DE TOULOUSE,, 18 December 2008 (2008-12-18), pages 1-205, XP007920183, abstract Sections 1.4.3, 2.3.1, 2.6.2, 6.4.2, 6.4.3 ----- -/--	1-10

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 June 2012	28/06/2012

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Almeida Garcia, B

2

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	PITOT: "The Voxar project", IEEE COMPUTER GRAPHICS AND APPLICATIONS, vol. 13, no. 1, 1 January 1993 (1993-01-01), page 27, XP055019008, ISSN: 0272-1716, DOI: 10.1109/38.180114 the whole document -----	1-10
A	"Out-Of-Core Algorithms for Scientific Visualization and Computer Graphics", COURSE NOTES FOR IEEE VISUALIZATION 2002, 1 October 2010 (2010-10-01), XP055019011, the whole document -----	1-10

2

RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°
PCT/FR2012/050784

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
INV. G06T15/00
ADD.

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
G06T

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si cela est réalisable, termes de recherche utilisés)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
X	VARADHAN G ET AL: "Out-of-core rendering of massive geometric environments", VIS 2002. IEEE VISUALIZATION 2002. PROCEEDINGS. BOSTON, MA, OCT. 27 - NOV. 1, 2002; [ANNUAL IEEE CONFERENCE ON VISUALIZATION], NEW YORK, NY : IEEE, US, 1 novembre 2002 (2002-11-01), pages 69-76, XP031212949, ISBN: 978-0-7803-7498-0 Sections 1 - 4.3, 6 et sous-sections -----	1-10
A	VIVANLOC V: "Rendu distribué sur grappe de CPU/GPU et effects d'éclairage global", THÈSE L'UNIVERSITÉ DE TOULOUSE,, 18 décembre 2008 (2008-12-18), pages 1-205, XP007920183, abrégé Sections 1.4.3, 2.3.1, 2.6.2, 6.4.2, 6.4.3 ----- -/--	1-10

[X] Voir la suite du cadre C pour la fin de la liste des documents

[ ] Les documents de familles de brevets sont indiqués en annexe

* Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent

"E" document antérieur, mais publié à la date de dépôt international ou après cette date

"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)

"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens

"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée

"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention

"X" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément

"Y" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier

"&" document qui fait partie de la même famille de brevets

Date à laquelle la recherche internationale a été effectivement achevée	Date d'expédition du présent rapport de recherche internationale
19 juin 2012	28/06/2012

Nom et adresse postale de l'administration chargée de la recherche internationale	Fonctionnaire autorisé
Office Européen des Brevets, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Almeida Garcia, B

2

C(suite). DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
A	PITOT: "The Voxar project", IEEE COMPUTER GRAPHICS AND APPLICATIONS, vol. 13, no. 1, 1 janvier 1993 (1993-01-01), page 27, XP055019008, ISSN: 0272-1716, DOI: 10.1109/38.180114 le document en entier	1-10
A	"Out-Of-Core Algorithms for Scientific Visualization and Computer Graphics", COURSE NOTES FOR IEEE VISUALIZATION 2002, 1 octobre 2010 (2010-10-01), XP055019011, le document en entier	1-10