FORMAT SDM 3.0

Préparé pour	
Préparé par	OKTAL 2,Rue Boudeville 31 100 Toulouse
Authentifié par OKTAL / R&D C. Giannesini Date: 04 / 08 / 95	**Approuvé par** Date: / /

Mots Clés

Structure de données, matériaux, motifs, appelBO, structure mémoire, structure fichiers.

Table des matières

Annexes

Sans objet

Figures

Sans objet

Tableaux

Sans objet

1 INTRODUCTION

Le but de ce document est de clairement expliciter la structure de donnée 3.0 (SDM), adaptée à la modélisation et utilisée par tous les produits OKTAL.

La structure de données SDM est composée de la description des données:

- Bdd: Définition des objets.
- Matériaux: Définition des matériaux.
- Correspondance motifs: Définition des motifs utilisés.
- Correspondance AppelBO: Définition des objets en bibliothèque.

Ce document traite exclusivement de la structure de donnée d'un point de vue interne (structures et tableaux de données en mémoire) et externe (format de fichier ascii). Les fonctions de manipulation de cette structure de données font l'objet d'un autre document.

Les descriptions mémoire et ascii des fichiers Bases de données (BDD), matériaux, de correspondances motifs et de description des objets en bibliothèque sont détaillées dans ce document.

2 REMARQUES PRÉALABLES:

Notion initiale:
Une base de données visuelle est constituée d'informations géométriques et de rendu. Ces informations sont contenues dans les fichiers **Bases de données (BDD)**.

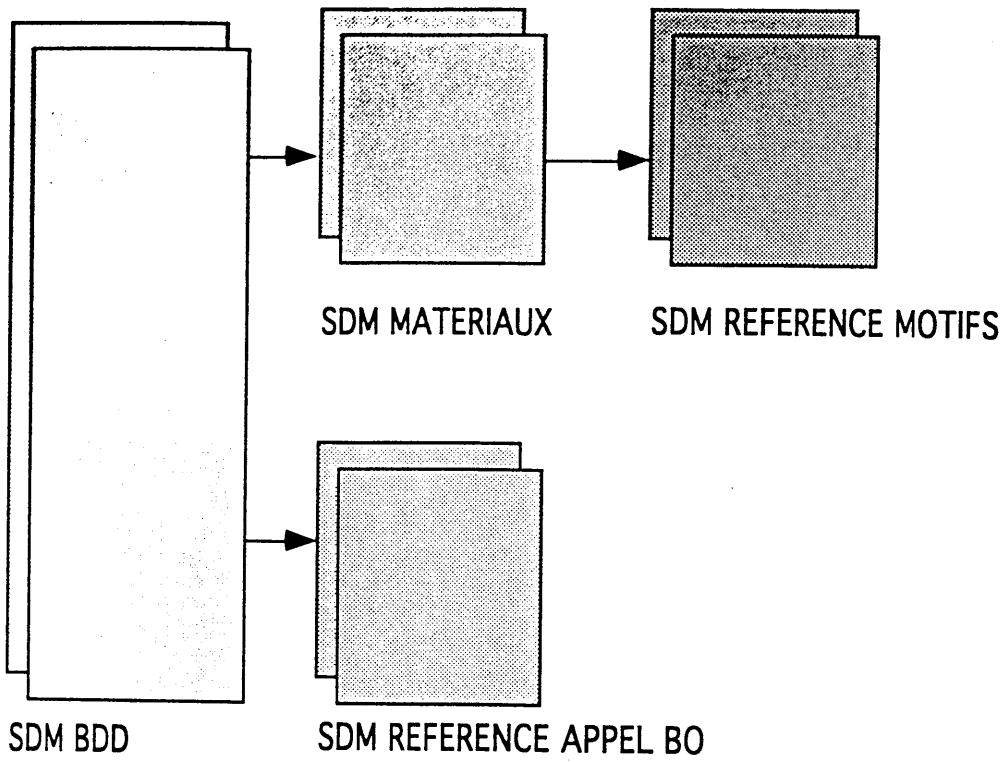
Les informations de rendu permettent d'indiquer les matériaux utilisés pour le rendu; un matériau est défini par des couleurs, une dureté, un motif de texture,... Les matériaux sont décrits dans des **fichiers matériaux**, qui pourront être partagés entre plusieurs Bases de données; de plus, si on désire modifier le rendu d'une base de donnée, il suffit de modifier les matériaux qu'elle utilise, ou de lui associer un autre fichier de matériaux. Le nom du fichier de matériaux utilisé par une base de donnée est indiqué dans le fichier de description de la base de données.

Dans une base de données, on peut référencer des objets se trouvant dans un bibliothèque, de façon à centraliser la représentation de ces objets. Ainsi, dans une base de données, on pourra définir plusieurs références à des objets en bibliothèque, en indiquant l'identificateur de l'objet référencé et sa position. Toutes les références sont regroupées dans un **fichier de définition des objets en bibliothèque**. Le nom du fichier utilisé par une base de données est indiqué dans le fichier de description de la base de données. Le fichier de définition des objets en bibliothèque contiendra, pour chaque référence, le nom du fichier base de données SDM, définissant ses informations géométriques et de rendu.

A chaque matériau est associé un motif de texture. La SDM est indépendante de la machine de visualisation. Chaque machine dispose de son propre format de définition des fichiers "image 2D". La correspondance entre le motif référencé par le matériau et le fichier "image 2D" du motif est faite dans un **fichier de correspondance des motifs**. Le nom du fichier utilisé par des matériaux est indiqué dans le fichier de définition des matériaux.

Le schéma suivant représente le lien entre tous les fichiers.

LIENS ENTRE LES SDM



Cette version de la SDM s'appuie sur quelques règles de base :

- Tout élément d'une base de données peut être soit un objet décrit globalement, soit un appel BO.
- Un appel BO ne peut pas être constitué d'appelBO.
- Les matériaux peuvent être partagés entre plusieurs fichiers bases de données.
- Les données de correspondance de motifs peuvent être partagées entre plusieurs fichiers matériaux.
- Les données de définition des objets en bibliothèque peuvent être partagées entre plusieurs fichiers bases de données.
- Tous les logiciels peuvent lire et écrire les fichiers ASCII, que ce soit en version française ou anglaise. La lecture est unique, alors que l'écriture se fait au choix d'une des deux versions possibles.
- Un champ sans aucun membre ou composé de valeurs par défaut n'est pas écrit dans un fichier ASCII.
- Des champs inexistants dans un fichier ASCII conduisent à des valeurs par défaut en mémoire.
- Tous les objets d'une même structure arborescente doivent être décrits dans un même fichier Bdd.
- Les indices de numérotation d'entités (sommets, faces, vecteurs, ...) commencent à 1 dans les fichiers ascii, alors qu'ils débutent à 0 en mémoire.
- Si le premier caractère d'une ligne est un # ou un @, la ligne est considérée comme un commentaire.

Possibilité de particulariser les fichiers ASCII:

Dans les fichiers ASCII, les champs sont repérés par des MOT CLEFS, défini de la façon suivante **<MOT CLEFS>**.

Les fichiers matériaux, correspondance motifs et définition des objets en bibliothèque accepte des champs utilisateur, définis pour une extension de structure de données. Les fonctions de lecture standards de ces fichiers ignorent ces champs utilisateur. L'utilisateur pourra alors réaliser une fonction de lecture supplémentaire permettant de prendre en compte ces champs. Cependant, les champs utilisateur pourront être rajoutés dans les définitions de chaque entité (MATERIAU, MOTIF, OBJETS), et ne pourront pas constituer une entité nouvelle.

Dans certains cas, un utilisateur peut vouloir réaliser des extensions plus importantes, en définissant des blocs entiers de définition de données (par exemple une rubrique supplémentaire ASPECT FACES pour les objets). Pour réaliser cela, on peut utiliser le fait que le caractère @ indique un commentaire. Ainsi, toutes les lignes débutant par @ sont ignorées par les fonctions de lecture standard. L'utilisateur pourra alors réaliser une fonction de lecture supplémentaire permettant de prendre en compte ces champs. Les mots clefs de l'utilisateur seront @MOT CLEF@.

Exemple

Rubrique standard:

```
<ASPECT>    2
        1   <MATERIAU>    10
            <TYPE>        2
```

Rubrique utilisateur:

```
@MY_ASPECT@
        @1  @COMPORTEMENT@  2
```

Pour particulariser les données géométriques SDM, il est également possible d'utiliser les mécanisme d'extension mis en place dans la librairie de fonctions de manipulation SDM.

Ces mécanismes permettent d'étendre la SDM grace à des définitions de fonctions (parmi lesquelles figurent les fonctions de lecture et d'écriture des extensions).
A partir de ce moment, toutes les fonctions de gestion de la SDM géreront les extensions de façon transparentes.

Recherche des fichiers ASCII:

Les noms des fichiers matériaux et de définition des objets en bibliothèque sont indiqués dans les fichiers de définition de Bases de Données. Le nom du fichier de correspondance des motifs est indiqué dans le fichier matériau.

Afin de faciliter la portabilité des bases de données entre plusieurs machines, ou de ne pas les contraindre à se trouver sous des arborescences de répertoires figées, une notion de recherche des fichiers par "pathname" a été introduite. Le principe en est le suivant:

Si le fichier dont le nom est indiqué existe, on le prend en compte,

Sinon on recherche son existence dans une série de répertoires (path) indiqué au travers de variables d'environnement.

La variable d'environnement permettant de définir les path de recherche des fichiers matériaux, de définition des objets en bibliothèque et de correspondance motifs se nomme

SDMPATH

Sa définition doit être faite selon la syntaxe suivante:

setenv SDMPATH repertoire1:repertoire2:...

De plus, dans les fichiers de correspondance motifs, on indique le nom des fichiers "image" des motifs utilisés. Dans le même soucis de portabilité, une variable de définition des répertoires de recherche des fichiers image a été introduite. Son principe de fonctionnement est le même que celui de SDMPATH.

Cette variable d'environnement se nomme

SDMPATHMOT

Sa définition doit être faite selon la syntaxe suivante:

setenv SDMPATHMOT repertoire1:repertoire2:...

Dans les fichiers de définition des objets en bibliothèque, on indique le nom des fichiers bases de données de description de chaque objet. Dans le même soucis de portabilité, une variable de définition des répertoires de recherche des fichiers de description bases de données a été introduite. Son principe de fonctionnement est le même que celui de SDMPATH.

Cette variable d'environnement se nomme

SDMPATHABO

Sa définition doit être faite selon la syntaxe suivante:

setenv SDMPATHABO repertoire1:repertoire2:...

Enfin, lors de l'écriture des fichiers ASCII, on offre la possibilité de maîtriser le chemin d'écriture. Par exemple, si un fichier matériau se nomme repertoire1/repertoire2/matériau, il sera possible d'indiquer ce nom tel quel dans le fichier Bdd qui utilise ces matériaux, mais également de modifier le chemin d'accès au fichier, au moment de l'écriture en forçant, par exemple, à écrire ./matériau.
Cette possibilité complète les notions de path définies ci dessus.
La maîtrise des écritures est faite au travers de variables d'environnement particulières permettant de définir les répertoires d'écriture. le principe de fonctionnement est le suivant.

Si la variable n'existe pas, le nom du fichier à écrire est inchangé,
Sinon on écrit le nom du fichier, précédé de la donnée définie dans la variable d'environnement

Les variables d'environnement disponibles sont les suivantes:

SDMPATHECRmat pour écriture des fichiers matériaux dans les fichiers Bases de données.
SDMPATHECRcbo pour écriture des fichiers de définition des objets en bibliothèque dans les fichiers Bases de données.
SDMPATHECRcmt pour écriture des fichiers de correspondance motifs dans les fichiers matériaux.
SDMPATHECRmot pour écriture des fichiers "images" des motifs dans les fichiers de correspondance de motifs.
SDMPATHECRabo pour écriture des fichiers de définition des objets dans les fichiers de définition des objets en bibliothèque.

Ces variables sont définies selon la syntaxe suivante:

setenv **SDMPATHECRmat** répertoire

Définition des types de tableaux:

ISET(typevar) var : permet de définir un pointeur "var" de type "typevar" (typevar *var). Le tableau géré est un tableau à trous.

ILISTE(typevar) var : permet de définir un pointeur "var" de type "typevar" (typevar *var). Le tableau géré est un tableau sans trous.

Définition de type:

INDICE_SDM permet de définir le type d'indice utilise. (char, short, unsigned short, int).

3 STRUCTURE DE DONNÉES MÉMOIRE BDD

3.1 Description d'un objet

Chaque objet est décrit de manière indépendante:

```
typedef struct  {
                TYPEOBJET       type_objet;             (1)
                short           num_objet;              (2)
                char            nom_objet[MAXNOMOBJ];   (3)

                short           instance;               (4)

                INDICE_SDM      nb_art;                 (5)
                INDICE_SDM      nb_nvd;                 (5)
                INDICE_SDM      nb_face;                (5)
                INDICE_SDM      nb_sommet;              (5)
                INDICE_SDM      nb_vecteur;             (5)
                INDICE_SDM      nb_position;            (5)
                INDICE_SDM      nb_materiau;            (5)
                INDICE_SDM      nb_matrice;             (5)

                PRESENCE        typ_lum;                (6)
                PRESENCE        typ_tri;                (6)
                PRESENCE        typ_tex;                (6)
                PRESENCE        typ_alias;              (6)
                PRESENCE        mat_ext;                (6)

                ILISTE(des_nvd) nvd;                    (7)
                ILISTE(des_art) art;                    (8)

                ISET(poly)      face;                   (9)
                ISET(norm)      normale;                (9)
                ISET(lum)       luminance;              (9)
                ISET(asp_face)  aspect_face;            (9)

                ISET(matériau)  mater;                  (10)
                ISET(matrice)   mat;                    (11)

                ISET(som)       sommet;                 (12)
                ISET(vect )     vecteur;                (12)
                ISET(asp_som)   aspect_som;             (12)

                ILISTE(posit)   position;               (13)
```

```
    ASSOCIE_PRIVE   *assoc;                 (14)
    short           derniere_extension;     (15)
    void            **tab_extension;        (15)
} des_objet;
```

(1). **Un élément Bdd peut être soit un objet défini complètement dans la structure des_objet, soit un appel BO, c'est à dire une référence via un fichier de définition vers un objet défini complètement dans un autre fichier SDM. Choix entre OBJET ou APPELBO. La valeur par défaut est *OBJET*.**

(2). **Il s'agit du numéro absolu d'un objet dans une Bdd. Ce champ permet d'indiquer les objets fils dans une définition d'arborescence. La valeur par défaut est *DEFAULT_NUM_OBJET* (-1).**

(3). **Ce champ permet à l'utilisateur d'identifier un objet par son nom. La valeur par défaut est *DEFAULT_NOM_OBJET* (SANS_NOM).**

(4). **Ce champ correspond à la référence d'un objet dans le fichier de correspondance, dans le cas des appels BO. La valeur par défaut est *DEFAULT_INSTANCE* (-1).**

(5). Toutes les valeurs *nb_xxx* définissent le nombre exact d'éléments associés à leurs objets respectifs.

(6). Toutes les variables du type PRESENCE sont des drapeaux validant ou pas la présence des propriétés associées :

typ_lum	->	ombrage ou pas. Valeur par défaut *DEFAULT_TYPELUM* (PRESENT) Avec ombrage
typ_tri	->	trie par priorité statique ou pas. Valeur par défaut *DEFAULT_TYPETRI* (PRESENT) Trié
typ_tex	->	objet texturé ou pas. Valeur par défaut *DEFAULT_TYPETEX* (PRESENT) Texturé
typ_alias	->	Aliasé ou pas. Valeur par défaut *DEFAULT_TYPEALIAS* (PRESENT) Antialiasé
mat_ext	->	Fichier matériau externe présent ou pas. Valeur par défaut *DEFAULT_REFMAT* (ABSENT) Interne

Ces drapeaux sont définis au niveau de l'objet; ils permettent d'ignorer les champs définies au niveau des faces, lorsqu'ils sont a ABSENT. Les valeurs possibles sont ABSENT ou PRESENT.

(7). Ce champ est le tableau de niveaux de détails associés à un objet donné. Il contient les numéros absolus des fils "*niveaux de détail*" de l'objet.

(8). Ce champ est le tableau des articulations constituant un objet donné. Il contient les numéros absolus des fils "*articulations*" de l'objet.

(9). Ces champs sont les pointeurs définissant les tableaux d'éléments de description liés aux faces d'un objet donné :

face	->	description géométrique des faces.
normale	->	description des normales aux faces.
luminance	->	descriptions des données pour ombrage des faces.
aspect_face	->	description des aspects liés aux faces.

(10). Il s'agit du tableau des matériaux INTERNES associés aux différentes faces ou aux différents points de l'objet.

(11). Définition des matrices de texture partageables par les faces de l'objet, sachant qu'il y a au maximum autant de matrices de texture que de faces. Cela permet d'appliquer la même matrice à plusieurs faces si besoin est.

(12). Ces champs sont les pointeurs définissant les tableaux d'éléments de description liés aux sommets d'un objet donné :

sommet	->	coordonnées des sommets.
vecteur	->	coordonnées des normales aux sommets
aspect_sommet	->	description des aspects liés aux sommets.

(13). Ce champ correspond au pointeur vers le tableau décrivant les informations de positionnement de l'objet.

(14). Ce champ est prive à la SDM. Il sera utilisé par la gestion des tables associées aux faces et aux sommets.

(15). Ces champs sont prive à la SDM. Il seront utilisés par la gestion des extensions de l'objet.

3.2 Structures internes

Toutes les structures internes sont celles utilisées dans la structure des_objet pour décrire un objet donné.

3.2.1 Définition des niveaux de détails

Afin d'optimiser la visualisation en temps réel, on associe à un objet donné plusieurs autres objets identiques, mais géométriquement dégradés (moins de faces).

```
typedef struct  { short     num_objet;      (1)
                  double    distmin;        (2)
                  double    distmax;        (3)
                } des_nvd;
```

(1) Numéro absolu d'objet de type *OBJET* ou de type *APPELBO* associé à un niveau de détail donné, défini dans le même fichier. Il s'agit des numéros absolu d'objet. La valeur par défaut est *DEFAULT_NUM_OBJET* (-1).

(2) Distance minimum de commutation de niveau de détail. La valeur par défaut est *DEFAULT_DIST_NVD* (0).

(3) Distance maximum de commutation de niveau de détail. La valeur par défaut est *DEFAULT_DIST_NVD* (0).

3.2.2 Définition des articulations

```
typedef struct  { short     type;           (1)
                  short     num;            (2)
                } des_art;
```

(1) Variable définissant la loi de déplacement de l'objet *articulation*. La valeur par défaut est *DEFAULT_TYPE_ART* (0).

(2) Numéro absolu de l'objet correspondant à l'articulation. La valeur par défaut est *DEFAULT_NUM_ART* (-1).

3.2.3 Définition des données face

Chaque face est décrite par un nombre de sommets dont les indices sont stockés dans un tableau.

```
typedef struct  { INDICE_SDM            nb_sommet;  (1)
                  ILISTE(INDICE_SDM)    ind_sommet; (2)
                } poly;
```

(1) Nombre exact de sommets constituant la face.
(2) Tableau contenant les indices absolus des sommets constituant la face. Ces indices sont ceux du tableau de définition des coordonnées des sommets de l'objet.

3.2.4 Définition des données des normales aux faces

A chaque face est associé un vecteur normal indiquant l'orientation de la face.

```
typedef struct  { double  nx;   (1)
                  double  ny;   (1)
                  double  nz;   (1)
                } norm;
```

(1) Coordonnées de la normale à la face. Les valeurs par défaut sont *DEFAULT_NORMALE_X, DEFAULT_NORMALE_Y, DEFAULT_NORMALE_Z* (0).

3.2.5 Définition des données d'ombrage

Pour les faces ombrées en Gouraud, on gère un tableau d'indices pointant vers les normales aux sommets.

```
typedef struct  { INDICE_SDM            nb_vecteur;  (1)
                  ILISTE(INDICE_SDM)    ind_vecteur; (2)
                } lum;
```

(1) Nombre exact de vecteurs constituant le tableau ind_vecteur.
(2) Tableau contenant les indices absolus des normales aux sommets constituant la face. Ces indices sont ceux du tableau de définition des coordonnées des normales aux sommets de l'objet.

3.2.6 Définition des données d'aspétisation des faces

A chaque face est associé un nombre de données d'aspétisation (pour le rendu) :

```
typedef struct {  TYPEFACE          type_face;      (1)
                  TYPE_OMBRAGE      type_omb;       (2)
                  TYPEASPECT        type_asp;       (3)
                  TYPETEXTURE       type_tex;       (4)
                  TYPEEXPLOIT       type_exp;       (5)
                  MODEVISFACE       mode_visu       (6)
                  PRESENCE          type_alias;     (7)
                  int               priorite        (8)
                  INDICE_SDM        ind_matrice;    (9)
                  short             trans;          (10)
                  short             groupe;         (11)
                  short             util;           (12)
                  short             codmat_face     (13)
                  short             codmat_intern   (14)
                  short             épaisseur;      (15)
               } asp_face;
```

(1) Renseigne sur le type de la face :
FACE_ORIENTEE (ou ORIENTEE),
FACE_NON_ORIENTEE (ou NON_ORIENTEE),
LIGNE_NON_ORIENTEE (ou LIGNE),
POINT_NON_ORIENTE (ou FPOINT),
FEU_NON_ORIENTE (ou FEU),
LIGNE_ORIENTEE,
POINT_ORIENTE,
FEU_ORIENTE.
La valeur par défaut est *DEFAULT_ASPECT_TYPE_FACE* (FACE_ORIENTEE).

Des définitions supplémentaires permettent de déduire de TYPEASPECT le type d'élèment et le type d'orientation d'un élément.

```
typedef enum  {  ELT_FACE,
                 ELT_LIGNE,
                 ELT_POINT,
                 AUCUN_ELT
              }   TYPEELT;
```

permet de renseigner sur le type d'élèment d'une face grâce aux fonctions:

```
Sdm3dLireTypeelt(bdd,obj,face)
ISET(des_objet)     bdd;
int                 obj;
int                 face;
```

et

```
Sdm3dLireTypeeltAspect(aspect)
asp_face            *aspect;
```

```
typedef enum  {  ELT_ORIENT,
                 ELTNON_ORIENT,
                 AUCUN_ORIENT
              }   TYPEORIENT;
```

permet de renseigner sur le type d'orientation d'une face grace aux fonctions:

```
Sdm3dLireOrientation(bdd,obj,face)
ISET(des_objet)     bdd;
int                 obj;
int                 face;
```

et

```
Sdm3dLireOrientationAspect(aspect)
asp_face            *aspect;
```

Ces fonctions sont définies sous forme de define de façon à optimiser les temps d'accès.

(2) Renseigne sur le type d'ombrage de la face :
SANS,
UNIFORME,
GOURAUD,
PHONG.
La valeur par défaut est *DEFAULT_ASPECT_TYPE_OMB* (UNIFORME).

(3) Renseigne sur le type de coloration appliquée à la face :
COLORE_SEUL (ou COLORE),
COLORE_TEXTURE (ou TEXTURE),
COULEUR_INTERPOLE,
TEXTURE_COULEUR_INTERPOLE,
TEXTURE_INTERPOLE,
TEXTURE_SEUL
La valeur par défaut est *DEFAULT_ASPECT_TYPE_ASP* (COLORE_TEXTURE).

(4) Renseigne sur le type de plaquage de la texture en cas de texturation de la face:
MANUEL,
GROUND_MAPPING,
PENTE,
AXE_PRINCIPALE,
DEUX_AXES,
AXE_PRINCIPAL_AJUSTE,
DEUX_AXES_AJUSTE_U,
DEUX_AXES_AJUSTE_V,
DEUX_AXES_AJUSTE_UV,
AXE_PRINCIPAL_DEVELOPPE,
DEUX_AXES_DEVELOPPE_U,
DEUX_AXES_DEVELOPPE_V,
DEUX_AXES_AJUSTE_U_DEVELOPPE_V,
DEUX_AXES_AJUSTE_V_DEVELOPPE_U,
DEUX_AXES_DEVELOPPE_UV

La valeur par défaut est *DEFAULT_ASPECT_TYPE_TEX* (GROUND_MAPPING).

(5) Renseigne sur la nature de la face:
AUCUN,
ROULEMENT ,
COLLISION.
La valeur par défaut est *DEFAULT_ASPECT_TYPE_EXP* (AUCUN).

(6) Indique le mode de visualisation de la face:
VISU_NORMAL,
VISU_POL2D
La valeur par défaut est *DEFAULT_ASPECT_MODE_VISU* (VISU_NORMAL).

(7) Indique si la face est aliasée ou non:
ABSENT,
PRESENT

La valeur par défaut est *DEFAULT_ASPECT_ALIAS* (PRESENT).

(8) Donne la priorite affectée à la face.
La valeur par défaut est *DEFAULT_PRIORITE* (0).

(9) Donne l'indice global de la matrice de texture associée à la face, dans le tableau des matrices de la structure des_objet, pour le cas où la face est texturée.
La valeur par défaut est:
DEFAULT_ASPECT_IND_MATRIX (INDICE_SDM_DEFAULT).

(10) Donne le niveau de transparence associé à la face (0 à 255; 255 : transparent).
La valeur par défaut est *DEFAULT_ASPECT_TRANS* (0).

(11) Donne le numéro de groupe associé à la face (nature thermique par exemple).
La valeur par défaut est *DEFAULT_ASPECT_GROUP* (-1).

(12) Champ laissé libre pour des besoins utilisateur.
La valeur par défaut est *DEFAULT_ASPECT_UTIL* (-1).

(13) Indice du matériau **externe** associé à la face dans le fichier de matériaux. il s'agit du numéro absolu de matériau utilisé pour la face.
La valeur par défaut est *DEFAULT_ASPECT_IND_MATER* (-1).

(14) Indice du matériau **interne** associé à la face lorsqu'il est défini dans *des_objet*
La valeur par défaut est *DEFAULT_ASPECT_IND_MATER _INT* (-1).

(15) Epaisseur de la face dans le cas où son type est soit LIGNE, POINT, ou FEU.
La valeur par défaut est *DEFAULT_ASPECT_EPAISSEUR* (1).

3.2.7 Définition des données d'aspétisation des sommets

Ces données seront utilisées pour l'interpolation de couleur sur une face.
A chaque sommet est associé un nombre de données d'aspétisation :

```
typedef struct  { short      codmat_som;     (1)
                  short      codmat_intern;  (2)
                } asp_som;
```

(1) Indice du matériau **externe** associé au sommet dans le fichier de matériaux. Il s'agit du numéro absolu de matériau utilisé pour le sommet.
La valeur par défaut est *DEFAULT_ASPECT_IND_MATER* (-1).
(2) Indice du matériau **interne** associé au sommet lorsqu'il est défini dans *des_objet*
La valeur par défaut est *DEFAULT_ASPECT_IND_MATER _INT* (-1).

3.2.8 Définition des matériaux liés aux faces et aux points

A chaque face, ou sommet on associe un matériau propre.

VOIR LA DESCRIPTION DE LA RUBRIQUE DANS LA DESCRIPTION MÉMOIRE DES MATÉRIAUX

3.2.9 Définition des sommets d'un objet

Tous les sommets d'un objet sont décrits dans le tableau "*sommet*" de la structure des_objet, chacun étant défini comme suit :

```
typedef struct  { double   x;      (1)
                  double   y;      (1)
                  double   z;      (1)
                } som;
```

(1) Coordonnées 3D du sommet.
Les valeurs par défaut sont *DEFAULT_COORD_X, DEFAULT_COORD_Y, DEFAULT_COORD_Z* (0).

3.2.10 Définition des vecteurs d'un objet

Les vecteurs d'un objet sont les normales aux sommets de l'objet. Ces normales sont utilisées pour l'ombrage de GOURAUD.
Le tableau des vecteurs est indépendant de celui des sommets. Un vecteur est associé à une face uniquement à travers la luminance d'une face. Un vecteur est décrit comme suit :

```
typedef struct  { double      lx;       (1)
                  double      ly;       (1)
                  double      lz;       (1)
                  INDICE_SDM  nb_ref;   (2)
                } vect;
```

(1) Coordonnées 3D du vecteur.
Les valeurs par défaut sont *DEFAULT_VECTEUR_X, DEFAULT_VECTEUR_Y, DEFAULT_VECTEUR_Z* (0).
(2) Nombre de références faites vers ce vecteur par les luminances faces.

3.2.11 Définition des positions d'un objet

Il peut être attribué plusieurs positions à un objet, chacune étant définie comme suit :

```
typedef struct  {  TYPEPOSITION    type_pos;     (1)
                   double          posx;         (2)
                   double          posy;         (2)
                   double          posz;         (2)
                   LIGMAT44        *matpos;      (2)
                }  posit;
```

(1) Définit le type de transformation à réaliser
TRANSLATION,
ROTATION,
ECHELLE,
MATRICIEL

La valeur par défaut est *DEFAULT_TYPE_POSITION* (TRANSLATION).
Il est possible de définir une position d'un objet par une matrice de position et/ou par des valeurs de translation,rotation ou echelle en X,Y et Z.

(2) Coordonnées 3D de la position de l'objet.
Translation en X, Y, Z.
Rotation autour de X, Y, Z.
Echelle en X, Y, Z.

Les valeurs par défaut sont *DEFAULT_POSIT_X*, *DEFAULT_POSIT_Y*, *DEFAULT_POSIT_Z* (0).

LIGMAT44 permet de définir une matrice de position 4x4, au format "américain"; les valeurs de translation sont sur la dernière ligne:

```
x    x    x    x
x    x    x    x
x    x    x    x
trx  try  trz  1
```

Par défaut, il n'y a pas de matrice de position (le pointeur est égal à NULL).

REMARQUE:
Les fonctions de la libsdm permettent de gérer les matrices de position. Pour cela il est essentiel de créer les "positions" au travers des fonctions suivantes:

Sdm3dEcrireTranslationPosition(ISET(des_objet)bdd, int objet, int num, double tabval[3])
Sdm3dEcrireRotationPosition(ISET(des_objet)bdd, int objet, int num, double tabval[3])
Sdm3dEcrireEchellePosition(ISET(des_objet)bdd, int objet, int num, double tabval[3])
Sdm3dEcrireMatricePosition(ISET(des_objet)bdd, int objet, int num, double mat[4][4])
On peut ensuite accéder aux champs de la matrice de position par mat[i][j].

3.2.12 Définition des matrices de textures

Chaque matrice est définie par ses coefficients et par un entier signifiant le nombre de faces la partageant.

```
typedef struct  { INDICE_SDM nb_face;                          (1)
                  double     uv[NB_LIGNES][NB_COLONNES];       (2)
                } matrice;
```

(1) Nombre de faces partageant la matrice .
(2) Coefficients de la matrice de plaquage de texture.
NB_LIGNES = 2 et NB_COLONNES = 4.

4 STRUCTURE DU FICHIER ASCII BDD

Il s'agit de la description des fichiers Bdd sous forme ASCII.

Les mot clefs sont indiqués en français ou en anglais.
Les valeurs entre {} indiquent des valeurs optionnelles.
On précise pour chaque champ les valeurs possibles; les valeurs *V* correspondent aux valeurs par défaut.
En lecture, si le champ est absent, la valeur par défaut est prise en compte.
En écriture, si à un champ est associée une valeur par défaut, ce champ n'apparaît pas dans le fichier.

 # Oktal Fichier : NomFichier Bdd Version 2.0
 # Oktal File : Filename Bdd Version 2.0

<NOM_BO> nom_fichier (*FIC_BO*)
<BO_NAM>

<NOM_MATER> nom_fichier (*FIC_MATER*)
<MATER_NAM>

<OBJET> num_objet (*-1*) {nom_objet (*SANS_NOM*)}
<OBJECT>

<NIVDETAIL> nb_nvd
<LOD>
1. num_objet1/appelBO1 dist1 dist2
2. num_objet2/appelBO2 dist1 dist2
.....
N num_objetN/appelBON dist1 dist2

<ARTICULATION> nb_art
<SUB_OBJECT>
1. num_objet1/appelBO1 type_deplacement
2. num_objet2/appelBO2 type_deplacement
.....
N num_objetN/appelBON type_deplacement

<TYPELUM> nb_typlum (Par défaut -> 0 : *2*)
<SHADED>
1. typelum (*2* : avec ombrage; 1 : sans ombrage)

<TYPETEX> nb_typtex (Par défaut -> 0 : *2*)
<TEXTURE>
1. typetex (*2* : avec texture; 1 : sans texture)

<TYPETRI> nb_typtri (Par défaut -> 0 : *2*)
<ORDERED>
1. typetri (*2* : trié; 1 : triable)

<TYPEANTIALISEE> nb_typeantialias (Par défaut -> 0 : *2*)
<ANTIALISEDTYP>
1. typeantialias (*2* : antialisé; 1 : absent)

<REFMAT> nb_refmat (Par défaut -> 0 : *1*)
<ATTR_FLAG>
1. typerefmat (2 : interne; *1* : externe)

<FACE> nb_face
<POLYGON>
1. nb_som som1 som2 som3 ...
.....
N nb_som som1 som2 som3 ...

<SOMMET> nb_som
<VERTEX>
1. X_1 Y_1 Z_1
.....
.....
N X_N Y_N Z_N

<NORMALE> nb_norm
<POLY_NORMAL>
ref_face_N $X_{ref\_face\_N}$ $Y_{ref\_face\_N}$ $Z_{ref\_face\_N}$
.....
(0 0 0)
.....
ref_face_M $X_{ref\_face\_M}$ $Y_{ref\_face\_M}$ $Z_{ref\_face\_M}$

<LUMINANCE> nb_lum
<LUMINANCE>
ref_face_N nb_vect ind_vect1 ...
.....
.....
ref_face_M nb_vect ind_vect1 ...

<VECTEUR> nb_vect
<VERT_NORMAL>
ref_som_N $X_{ref\_som\_N}$ $Y_{ref\_som\_N}$ $Z_{ref\_som\_N}$
.....
(0 0 0)
.....
ref_som_M $X_{ref\_som\_M}$ $Y_{ref\_som\_M}$ $Z_{ref\_som\_M}$

<PRIORITE> nb_prio
<PRIORITY>
ref_face_N prio_face_N *(0)*
.....
.....
ref_face_M prio_face_M *(0)*

<ASPECT_SOMMET> nb_asp_som
<VERT_ATTR>
ref_som_N <CODMAT> ref_mat_som
<ATTR_REF> *(-1)*
<CODMATINT> ref_mat_som_int
<INT_REF_ATTR> *(-1)*

.....

.....

ref_som_M **<CODMAT>** ref_mat_som
<ATTR_REF> *(-1)*
<CODMATINT> ref_mat_som_int
<INT_REF_ATTR> *(-1)*

<ASPECT_FACE> nb_asp_face
<POLY_ATTR>
ref_face_N **<TYPE>** typeface {épaisseur (*1*) }
<TYPE>
1 FACE_ORIENTEE ou ORIENTEE
2 FACE_NON_ORIENTEE ou NON_ORIENTE
3 LIGNE_NON_ORIENTEE ou LIGNE
4 POINT_NON_ORIENTE ou FPOINT
5 FEU_NON_ORIENTE ou FEU
6 LIGNE_ORIENTEE
7 POINT_ORIENTE
8 FEU_ORIENTE

<OMBRAGE> typeombrage
<SHADING>
1 SANS
2 UNIFORME
3 GOURAUD
4 PHONG

<ASPECT> typeaspect
<ASPECT>
1 COLORE_SEUL ou COLORE
2 COLORE_TEXTURE ou TEXTURE
3 COULEUR_INTERPOLE
4 TEXTURE_COULEUR_INTERPOLE
5 TEXTURE_INTERPOLE
6 TEXTURE_SEUL

<TEXTURE> typeprojection (Si aspect = TEXTURE)
<TEXTURE>
1 MANUEL
2 GROUND_MAPPING
3 PENTE
4 AXE_PRINCIPAL
5 DEUX_AXES
6 AXE_PRINCIPAL_AJUSTE
7 DEUX_AXES_AJUSTE_U
8 DEUX_AXES_AJUSTE_V
9 DEUX_AXES_AJUSTE_UV
10 DEUX_AXES_DEVELOPPE
11 DEUX_AXES_DEVELOPPE_U
12 DEUX_AXES_DEVELOPPE_V
13 DEUX_AXES_AJUSTE_U_DEVELOPPE_V
14 DEUX_AXES_AJUSTE_V_DEVELOPPE_U
15 DEUX_AXES_DEVELOPPE_UV

<EXPLOITATION> typeexploitation
<EFFECT>
1 AUCUN
2 ROULEMENT
3 COLLISION

<MODE_VISU> facenormal ou pol2d
<VIS_MOD>
1 VISU_NORMAL
2 VISU_POL2D

<CODMATFACE>	ref_mat_face_ext
<POLY_REF_ATTR>	*(-1)*
<CODMATINT>	ref_mat_face_int
<INT_REF_ATTR>	*(-1)*
<TRANSPARENCE>	transparence (255 : transparent)
<TRANSPARENCY>	*(0)*
<GROUPE>	valeur
<GROUP>	*(-1)*
<UTILISATEUR>	valeur
<USER>	*(-1)*
<MATRICETEX>	indice
<TEXMATRIX>	*(-1)*
<ANTIALIASEE>	(défaut -> 0 : 2)
<ANTIALIASED>	(2 : present; 1 : absent)

<MATRICE> nb_matrice (matrices de dimensions [2][4])
<MATRIX>

1.	nb_face_1	$coef_{00}$	$coef_{01}$	$coef_{02}$	$coef_{03}$
		$coef_{10}$	$coef_{11}$	$coef_{12}$	$coef_{13}$
.....					
.....					
N	nb_face_N	$coef_{00}$	$coef_{01}$	$coef_{02}$	$coef_{03}$
		$coef_{10}$	$coef_{11}$	$coef_{12}$	$coef_{13}$

<POSITION> nb_transfo
<PLACEMENT>

1.	typetransfo	X	Y	Z
.....				
	(***<TRANSLATION>***	*0*	*0*	*0*)
	(**<MAT_POSITON>**	x	x	x x)
	(	x	x	x x)
	(	x	x	x x)
	(	x	x	x x)
.....				
N	typetransfo	X	Y	Z

typetransfo **<TRANSLATION>** ou ***<TRANSLATION>***
<ROTATION> ou ***<ROTATION>***
<ECHELLE> ou ***<SCALE>***
<MAT_POSITION> ou **<PLACEMENT_SCALE>**

<DONNEES> nom_extension_francais
<DATA> nom_extension_anglais
Format spécifique des données de l'extension
<FIN_DONNEES> nom_extension_francais
<END_DATA> nom_extension_anglais
(Autant de paragraphe <DONNEES> que d'extensions définies pour l'objet)

<MATERIAU> nb_materiau

<ATTRIBUTE>

VOIR LA DESCRIPTION DE LA RUBRIQUE DANS LA DESCRIPTION DU FICHIER ASCII MATERIAU

<APPELBO> num_appelbo (*-1*) {nom—appelbo (*SANS_NOM*) }
<INSTANCE>

<ID> num_instance (*-1*)
<ID>

<POSITION> nb_transfo
<PLACEMENT>

1. typetransfo X Y Z

.....

(*<TRANSLATION>* *0 0 0*)
(<MAT_POSITON> x x x x)
(x x x x)
(x x x x)
(x x x x)

.....

N typetransfo X Y Z

typetransfo **<TRANSLATION>** ou *<TRANSLATION>*
<ROTATION> ou *<ROTATION>*
<ECHELLE> ou *<SCALE>*
<MAT_POSITION> ou **<PLACEMENT_SCALE>**

5 STRUCTURE DE DONNÉES MEMOIRE MATERIAUX

```
typedef struct {  coul     couleur;                  (1)
                  coul     ambiant;                  (1)
                  coul     emission;                 (1)
                  coul     specular;                 (1)
                  float    shininess;                (1)
                  short    alpha;                    (1)
                  short    num_mater;                (2)
                  short    num_motif;                (3)
                  short    durete;                   (4)
                  short    codtherm;                 (5)
                  float    ku;                       (6)
                  float    kv;                       (6)
                  char     nom_mat[MAXNOMMAT];       (7)
               } materiau;
```

(1) Renseigne sur les couleurs associées à la face, définie par le code RVB correspondant dans la structure coul :

```
typedef struct {
                    unsigned char  R;
                    unsigned char  V;
                    unsigned char  B;
                } coul;
```

Les valeurs par défaut des R, V, B sont *DEFAULT_R* (255), *DEFAULT_V* (255), *DEFAULT_B* (255) pour la couleur "couleur" et *NULL_R* (0), *NULL_V* (0), *NULL_B* (0) pour les autres couleurs .

couleur permet de préciser la couleur diffuse du matériau (couleur de base)
ambiant permet de préciser la couleur ambiante du matériau.
emission permet de préciser la couleur d'emissivité du matériau.
specular permet de préciser la couleur spéculaire du matériau.

shininess est le coefficient spéculaire du matériau:
La valeur par défaut est *DEFAULT_SHININESS* (0).

alpha permet d'indiquer le valeur de transparence du matériau (0 à 255; 255 transparent)
La valeur par défaut est *DEFAULT_ALPHA* (0).

(2) Numéro absolu du matériau.Ce numéro sera utilisé au niveau de la définition des aspects faces.
La valeur par défaut est *DEFAULT_NUM_MATERIAU* (-1).

(3) Renseigne sur le numéro motif associé à la face dans la table de correspondance de motifs.
La valeur par défaut est *DEFAULT_MATER_IND_MOTIF* (-1).

(4) Renseigne sur la dureté du matériau associé à la face.
La valeur par défaut est *DEFAULT_DURETE* (0).

(5) Renseigne sur le code thermique du matériau associé à la face.
La valeur par défaut est *DEFAULT_CODTHERM* (-1).

(6) Densités des motifs de textures.
Ceci permet de disposer de 2 matériaux partageant le même motif mais avec des dilations différentes.
Les valeurs par défaut sont *DEFAULT_KU* et *DEFAULT_KV* (1)

(7) Nom associé au matériau.
La valeur par défaut est *DEFAULT_NOM_MATER* (SANS_NOM).

6 STRUCTURE DU FICHIER ASCII MATERIAUX

 # Oktal Fichier : NomFichier Mat Version 2.0
 # Oktal File : Filename Mat Version 2.0

<NOM_MOTIF> nom_fichier *(NOMFIC_MOTIF)*
<MOTIF_NAM>

<MATERIAU> nb_materiau
<ATTRIBUTE>

ref_mat_N	<NOM>	nom_mat
(n° absolu)	*<NAME>*	*(SANS_NOM)*
	<COULEUR>	R V B
	<COLOR>	*(255 255 255)*
	<COUL_AMBIANTE>	R V B
	<AMBIANT_COLOR>	*(0 0 0)*
	<COUL_EMISSIVE>	R V B
	<EMISSIVE_COLOR>	*(0 0 0)*
	<COUL_SPECULAIRE>	R V B
	<SPECULAR_COLOR>	*(0 0 0)*
	<TAUX_REFLEXION>	valeur
	<SHININESS>	*(0)*
	<TRANSMATER>	valeur (0 à 255; 255 transparent)
	<ALPHA>	*(0)*
	<CODMOTIF>	code du motif
	<MOTIF_REF>	*(-1)*
	<DURETE>	valeur
	<HARDNESS>	*(0)*
	<CODTHERM>	code nature thermique
	<THERM_REF>	*(-1)*
	<DENSITE>	ku kv
	<DENSITY>	*(1 1)*

Remarque:
Il est possible de rajouter des champs utilisateur dans la définition d'un matériau. Ces champs sont ignorés par les fonctions standards de lecture.

7 STRUCTURE DE DONNÉES MEMOIRE CORRESPONDANCE DE MOTIFS

```
typedef struct {  char            nom_motif[MAXSTRING];   (1)
                  char            nomfic_motif[MAXSTRING]; (2)
                  double          dimension;              (3)
                  short           num_motif;              (4)
                  int             id_texture              (5)
                  MODETEXTURE     mode_u                  (6)
                  MODETEXTURE     mode_v                  (7)
                  QUALITETEXTURE  qualite_texture         (8)
               } des_motif;
```

(1). Nom du motif.
La valeur par défaut est *DEFAULT_MOTIF_NOM* (NOM_MOTIF)
(2). Nom du fichier contenant la description du motif (image 2D).
La valeur par défaut est *DEFAULT_NOMFIC_MOTIF* (NOMFIC_MOTIF)
(3.) Dimension du motif.
La valeur par défaut est *DEFAULT_DIMENSION* (1)
(4). Numéro absolu du motif. Il sert à référencer un motifs pour un matériau donné.
La valeur par défaut est *DEFAULT_NUM_MOTIF* (-1)
(5). Identificateur de texture (N° de texture chargé). Ce champ sert éventuellement pour les logiciels de visualisation.
La valeur par défaut est *DEFAULT_ID_TEXTURE* (-1)
(6). Mode d'application du motif de texture dans la direction U.
Le type MODETEXTURE est défini comme suit :

```
typedef enum {
    SDM_REPEAT=1,    /* répéter le motif */
    SDM_CLAMP,       /**/
    SDM_SELECT       /**/
        } MODETEXTURE;
```

Ce mode permet de définir comment les coordonnées de texture n'appartenant pas à l'espace [0,1] sont prises en compte.
Le mode SDM_REPEAT répéte le motif de texture.
Le mode SDM_CLAMP ne répète pas le motif ; les pixels hors du motif prennent la couleur du bord du motif.
Le mode SDM_SELECT ne répète pas le motif ; ne traite pas les pixels hors du motif
La valeur par défaut est *DEFAULT_MODE_TEX* (SDM_REPEAT)
(7). Mode d'application du motif de texture dans la direction V.
La valeur par défaut est *DEFAULT_MODE_TEX* (SDM_REPEAT)

(8). Mode de codage des motifs de texture
Le type QUALITETXTURE est défini comme suit :

```
typedef enum {
    QUALITE_NORMAL=1,
    QUALITE_HAUTE
        } MODETEXTURE;
```

Ce mode permet de définir comment sont codées les motifs de texture, pour un GIS.

QUALITE_NORMAL permet d'indiquer que le codage est le codage standard de la machine (codage 16 bits sur les Silicon).

QUALITE_HAUTEpermet d'indiquer que le codage est le codage haute résolution de la machine (codage 32 bits sur les Silicon).

8 STRUCTURE DU FICHIER ASCII CORRESPONDANCE DE MOTIFS

 # Oktal Fichier : NomFichier Mtf Version 2.0
 # Oktal File : Filename Mtf Version 2.0

<MOTIF> nb_motif
<PATTERN>

ref_motif (*n° absolu*)	<NOM>	nom_motif	(*NOM_MOTIF*)
	<NAME>		
	<NOM_MOTIF>	nomfic_motif	(*NOMFIC_MOTIF*)
	<MOTIF_NAM>		
	<DIMENSION>	dimension	(*1*)
	<SIZE>		
	<MODETEX_U>	mode_u	
		1 SDM_REPEAT	
		2 SDM_CLAMP	
		3 SDM_SELECT	
	<MODETEX_S>		
	<MODETEX_V>	mode_v	
		1 SDM_REPEAT	
		2 SDM_CLAMP	
		3 SDM_SELECT	
	<MODETEX_T>		
	<QUALITE_TEXTURE>	qualite	
		1 QUALITE_NORMAL	
		2 QUALITE_HAUTE	
	<TEXTURE_QUALITY>		

Remarque:
Il est possible de rajouter des champs utilisateur dans la définition d'un motif. Ces champs sont ignorés par les fonctions standards de lecture.

9 STRUCTURE DE DONNÉES MEMOIRE CORRESPONDANCE D'OBJETS

```
typedef struct { char   nom_objet[MAXNOMOBJ];          (1)
                 char   nomfic_objet[MAXSTRING];       (2)
                 char   nomfic_objet_simple[MAXSTRING]; (3)
                 short  num_objet;                     (4)
               } des_liens_objets;
```

(1). Nom d'objet.
La valeur par défaut est *DEFAULT_OBJET_NOM* (NOM_OBJET).
(2). Nom du fichier contenant la description SDM de l'objet.
La valeur par défaut est *DEFAULT_NOMFIC_OBJET* (NOMFIC_OBJET).
(3). Nom du fichier contenant la description simplifiée de l'objet.
(permet de représenter l'objet avec moins de faces)
(l'une ou l'autre des 2 représentations sera utilisée)
(en fonction de la valeur de 2 ou de 3)
La valeur par défaut est *DEFAULT_NOMFIC_OBJET_SIMPLE* (NOMFIC_OBJET_SIMPLE).
(4). Numéro absolu de la correspondance objet. Ce numéro sert à référencer un objet en bibliothèque dans le fichier BDD.
La valeur par défaut est *DEFAULT_NUM_OBJET_BO* (-1).

10 STRUCTURE DU FICHIER ASCII CORRESPONDANCE D'OBJETS

```
# Oktal Fichier : NomFichier  Obj Version 2.0
# Oktal File : Filename  Obj Version 2.0

<OBJETS>                          nb_liens
<OBJECTS>

    ref_obj     <NOM>                      nom_objet   (NOM_OBJET )
    (n° absolu) <NAME>
                <NOM_OBJET>                nomfic_objet (NOMFIC_OBJET )
                <OBJECT_NAM>
                <NOM_OBJET_SIMPLE>  nomfic_objet (NOMFIC_OBJET_SIMPLE )
                <SIMPLE_OBJECT_NAM>
```

Remarque:
Il est possible de rajouter des champs utilisateur dans la définition d'un objet en bibliothèque. Ces champs sont ignorés par les fonctions standards de lecture.

11 DEFINITIONS SUPPLEMENTAIRES

```
typedef enum { FRANCAIS,
               ANGLAIS,
               AUTRE
             } TYPEVERSION;
```

Permet d'indiquer le type de version pour l'écriture des fichiers ASCII.