

OVNI V5

*Outil de Visualisation Interactif*

Manuel utilisateur et d’Installation



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Rédacteur | Vérificateurs | | Approbateur |
| Fonction | Responsable Produit Logiciel | Responsable Produit Technique | Utilisateur Logiciel | Directeur du DOTA |
| Nom | G. DURAND |  | E. COIRO | A.Roblin |
| Visa | Signature_GD |  |  |  |

GEN-F102-4 (GEN-SCI-017)

|  |  |  |
| --- | --- | --- |
| HISTORIQUE | | |
| Version Révision | Date de mise en application | Cause et/ou nature de l’évolution |
| 1.0 | 12/11/2010 | Création |
| 1.1 | 14/12/2010 | Ajout des descriptions de 2 nouveaux boutons : Groupes et Matériaux + corrections. Modifications dans Tcl et C pour s’adapter à ToGL V2.0 + ajouts d’Enregistrer Images par Tcl, formats Jpeg, png et ppm. |
| 1.2 | 05/01/2011 | Quelques corrections mineures et ajout d’une référence à glutming.zip pour une installation Windows.  Mise à jour des sources pour actualiser la lecture des fichiers .3ds avec la version 2.0 de lib3ds.  Les librairies statiques installées par défaut sont maintenant celles pour Linux. |
| 1.3 | 26/04/2011 | Ajout d’une précision sur la création de lien Linux vers un répertoire.  Possibilité de compiler une version dll appelable directement par un code Tcl/Tk. Possibilité de créer une version autonome, sans installation annexe, sous Windows et Linux. |
| 1.4 | 03/05/2011 | Ovni passe en Version 4 suite aux changements de structure des répertoires et à la mise au point des nouvelles versions dll et tclkit (abandon en cours de mktclapp car est incompatible avec les dernières versions de Tcl/Tk). |
| 1.5 | 04/05/2011 | Quelques corrections mineures de la documentation. |
| 1.6 | 06/05/2011 | La version dll fonctionne sur Sun Solaris. Ajout de fichiers d’aide à la génération sous Solaris (00\_pre\_build.sh) + corrections correspondantes dans la documentation. |
| 1.7 | 10/05/2011 | Ajout d’organigrammes en annexe C |
| 1.8 | 13/12/2013 | Ajout de la fonction de sauvegarde de fichiers .bak |
| 2.0 | 10/04/2019 | Réécriture de l’ensemble motivée par le changement de la bibliothèque logicielle graphique |
| 2.1 | 07/03/2020 | Compléments de descriptions de certaines fonctionnalités améliorées de lecture des fichiers 3ds et obj. |
| 2.2 | 25/04/2020 | Enregistrer sous… peut travailler sur des fichiers .bdd (par défaut) mais aussi sur des .obj, .g3d, .off |
| 2.3 | 25/05/2020 | Amélioration dans le dialogue Modifications, meilleure gestion des re-générations de listes OpenGL. |
| 2.4 | 3/06/2020 | Ajout d’une option pour forcer un recalcul des normales aux sommets sur des facettes déclarées comme planes. |
| 2.5 | 4/06/2020 | Ajout d’un tableau comparatif des formats 3D |
| 2.6 | 20/06/2020 | Ajout d’une annexe expliquant commet modifier un code généré par wxSmith pour supporter davantage de cartes graphiques |
| 2.7 | 14/10/2020 | Ajout d’une fonction permettant de fusionner plusieurs objets en 1 seul. L’ancien menu Fusionner est remplacé par « Ajouter une Bdd » plus conforme à ce qui est effectivement réalisé. |
| 2.8 | 04/11/2020 | Ajout d’un support des fichiers Bdd texte encodés en utf-8 ou en Ansi, du moins pour les formats SDM et Wavefront obj. |
| 2.9 | 16/11/2020 | Mise à jour mineure : la création de facettes à partir de points de différents objets est possible depuis quelque temps déjà, mais la doc disait toujours que ce n’était pas encore possible ! |
| 2.10 | 27/11/2020 | Améliorations du changement de sens de parcours des facettes |
| 2.11 | 30/11/2020 | Amélioration de la gestion de la fenêtre de manipulation par translation. Ajout des min et max sur les 3 axes dans les propriétés. |
| 2.12 | 17/12/2020 | Manipulation par changement d’échelle : 3 chiffres décimaux au lieu de 2. Validation par la touche « Entrée » au clavier  Ajout d’un raccourci clavier « n » ⬄ Sens des normales |
| 2.13 | 05/01/2021 | Ajout du décodage et de la sauvegarde dans les formats .stl Ascii et binaires |
| 2.14 | 23/01/2021 | Ajout du décodage partiel de fichiers polygones .ply (ou Stanford Polygon Format). Formats Ascii et binaires (little et big Endian). |
| 2.15 | 16/02/2021 | Ajout de la reconnaissance des tristrips (données spécifiques de facettes en triangles) dans les ply binaires. |
| 2.16 | 27/09/2021 | Ajout d’une primitive Tore. |
| 2.17 | 16/11/2021 | Paramétrage du nombre de threads pour OpenMP |
| 2.18 | 29/11/2021 | Paramétrage de la détection des doublons d’arêtes |
| 2.19 | 10/07/2022 | Références plutôt à la version wxWidgets 3.2.0 |
| 2.20 | 20/09/2022 | Ajout d’un choix de tailles d’icônes dans la barre d’outils |
| 2.21 | 10/10/2022 | Icônes au format .svg (scalable vector graphics) : format de fichier indépendant de la taille des icônes |

sommaire

[1 objet 5](#_Toc108371381)

[2 domaine d’application 6](#_Toc108371382)

[2.1 Historique du développement. 6](#_Toc108371383)

[2.2 Fonctionnalités principales du code 8](#_Toc108371384)

[3 documents de rÉfÉrence 10](#_Toc108371385)

[4 dÉfinitions et abrÉviations 11](#_Toc108371386)

[5 Interface du logiciel OVNI 11](#_Toc108371387)

[5.1 Fenêtre principale 11](#_Toc108371388)

[5.1.1 Mouvements de base à la souris 13](#_Toc108371389)

[5.1.2 Mouvements de base au clavier 13](#_Toc108371390)

[5.1.3 Les menus 14](#_Toc108371391)

[5.1.3.1 Le menu Fichier 15](#_Toc108371392)

[5.1.3.2 Le menu Affichage 20](#_Toc108371393)

[5.1.3.3 Le menu Primitive 22](#_Toc108371394)

[5.1.3.4 Le menu Repérage 24](#_Toc108371395)

[5.1.3.5 Le menu Image 26](#_Toc108371396)

[5.1.3.6 Le menu Déplacement 26](#_Toc108371397)

[5.1.3.7 Le menu Outils 27](#_Toc108371398)

[5.1.3.8 Le menu Transformations 38](#_Toc108371399)

[5.1.3.9 Le menu Options 39](#_Toc108371400)

[5.1.3.10 Le menu Aide 39](#_Toc108371401)

[5.1.4 Les boutons sous la barre de menus 40](#_Toc108371402)

[5.2 Les raccourcis au clavier 41](#_Toc108371403)

[5.3 Les menus contextuels 43](#_Toc108371404)

[6 Installation 43](#_Toc108371405)

[6.1 Exécutable Windows 43](#_Toc108371406)

[6.1.1 Remarques 1 44](#_Toc108371407)

[6.1.2 Remarques 2 44](#_Toc108371408)

[6.2 Librairies diverses 45](#_Toc108371409)

[6.2.1 OpenGL 45](#_Toc108371410)

[6.2.2 Glut 45](#_Toc108371411)

[6.2.3 Lib3ds 45](#_Toc108371412)

[6.2.4 Expat 46](#_Toc108371413)

[7 Compilation d’OVNI 46](#_Toc108371414)

[Annexe A - Format des fichiers 3D 49](#_Toc108371415)

[A.1 -  Généralités 49](#_Toc108371416)

[A.2 -  Fichiers SDM Oktal-SE (.bdd) 50](#_Toc108371417)

[A.3 -  Fichiers Object File Format (.off) 53](#_Toc108371418)

[A.4 -  Fichiers Wavefront Obj (.obj) 54](#_Toc108371419)

[A.5 -  Fichiers Autodesk (.3ds) 56](#_Toc108371420)

[A.6 -  Fichiers Polygon Files de Niratam (.ply) 57](#_Toc108371421)

[A.7 -  Fichiers Polygon Files de l’université de Stanford (.ply) 58](#_Toc108371422)

[A.8 -  Fichiers Milkshape 3D (.m3d) 60](#_Toc108371423)

[A.9 -  Fichiers du Groupe 3D (.g3d) 60](#_Toc108371424)

[A.10 -  Fichier .STL 62](#_Toc108371425)

[A.11 -  Tableau comparatif des contenus des fichiers 3D 65](#_Toc108371426)

[Annexe B - Calcul des normales dans OVNI 66](#_Toc108371427)

[B.1 -  Calcul des normales au barycentre des facettes 66](#_Toc108371428)

[B.2 -  Calcul des normales aux sommets des facettes 66](#_Toc108371429)

[Annexe C - Création d’un nouveau panneau sous wxSmith 74](#_Toc108371430)

[Annexe D - Compilation de wxWidgets 76](#_Toc108371431)

[Annexe E - Modification du code généré par wxSmith 78](#_Toc108371432)

# objet

Ce document décrit le mode d’utilisation et d’installation du code **OVNI** (**O**util de **V**isualisation et de modificatio**N** **I**nteractif) dans la version V5.0 datée du 10/04/2019. Ce code est utilisé afin de visualiser, et adapter à une utilisation dans le logiciel Onera Crira (**C**alcul du **R**ayonnement **I**nfra**R**ouge des **A**vions), des avions 3D au format Oktal SDM (extension de fichier .bdd). Ce code est également capable d’importer des objets 3D sous d’autres formats, comme Wavefront obj (.obj), Autodesk (.3ds), Milkshape 3D texte (.m3d), Polygon File de Niratam (.ply), Polygon File de l’université de Stanford (extension .ply également), Object File Format (.off), Groupe 3D (.g3d), Stéréolithographique (\*.stl) dans ses 2 variantes, Ascii et binaire.

Préambule :

Jusqu’à la version 4, OVNI s’appuyait sur Tcl/Tk, un langage de scripts, afin de créer et gérer l’interface graphique. Tcl/Tk est disponible sur de nombreuses plates-formes, dont Windows, Linux, Sun Solaris, sur lesquelles OVNI a été implanté et testé à l’Onera. Pour une simple utilisation d’OVNI, les éléments indispensables de Tcl/Tk étaient fournis. Au cas où l’utilisateur veut pouvoir modifier le code, il lui fallait alors installer une version complète de Tcl/Tk, gratuite, ainsi que des librairies annexes (lecture de fichiers 3ds, lecture de fichier g3d, intégration d’une fenêtre OpenGL dans Tcl/Tk).

En standard, un script Tcl/Tk se lance via un utilitaire de Tcl/Tk qui s’appelle wish. Wish interprète les scripts et si une fonction n’existe pas en standard dans Tcl/Tk ou n’a pas été intégrée dans les scripts, notamment pour des raisons de performances, l’utilisateur doit fournir une librairie dynamique (fichier .dll sous Windows ou .so sous Linux). C’est via cette librairie que peuvent être exécutées les extensions. Dans ce cas, l’utilisateur doit écrire et compiler le code de cette librairie : c’est ce qui a été fait jusqu’à la version 4 d’Ovni, à partir d’un code écrit en C. Un utilitaire annexe, mktclapp, encapsulait les scripts et la librairie Ovni dans un fichier exécutable (.exe sous Windows). L’intérêt pour l’utilisateur final était de s’affranchir d’une installation complète de Tcl/Tk. Comme rappelé précédemment, les éléments indispensables de Tcl/Tk étaient fournis avec l’exécutable, dans des sous-répertoires séparés, conjointement aux autres librairies annexes, et de ce fait, l’utilisateur n’avait qu’à lancer cet exécutable sans se soucier de l’infrastructure sous-jacente.

Depuis la version 3.32, il était possible de se passer d’une installation même partielle de Tcl/Tk, du moins sous Windows et Linux. Les exécutables spécifiques comportaient en interne tous les composants nécessaires, à l’exception d’une librairie dynamique (Togl20.dll sous Windows et Togl20.so sous Unix/Linux). C’est cette librairie qui fournit une interface vers OpenGL à Tcl/Tk. L’inconvénient de ces exécutables, construits par tclkit, est qu’ils étaient beaucoup plus gros que ceux s’appuyant sur une installation de Tcl/Tk, mais leur gros avantage est qu’ils sont autonomes. La version 4 n’était pas fonctionnellement différente de la version 3.32, mais la structure des répertoires, en particulier ceux utilisés pour générer les exécutables, avait été revue complètement.

Depuis la version 8.5 de Tcl/Tk (distribution ActiveTcl), et d’avantage avec la version 8.6, des incompatibilités sont apparues, en particulier avec le logiciel intermédiaire, mktclapp, qui permettait d’encapsuler un exécutable Windows (ou Linux). Ce logiciel utilise des fonctions déclarées obsolètes dans les dernières versions de Tcl/Tk et surtout, n’est plus maintenu par son auteur depuis plusieurs années. D’autre part, l’utilisation de tclkit, qui n’a pas besoin de mktclapp, et qui aurait été une alternative intéressante, s’avère un peu délicate, pas toujours très stable : en particulier on a pu constater des différences de comportement entre 2 machines, a priori au même niveau d’operating system et de mises à jour : plutôt gênant, même si à chaque fois, une parade a été trouvée.

Ainsi, en 2014, on s’est posé la question de revoir complètement la partie interface en s’affranchissant de Tcl/Tk. Le choix s’est porté vers la librairie wxWidgets (version 2.8 au départ du projet). C’est une librairie comportant tous les éléments qui intéressent Ovni, et dont le développement est toujours très actif. Elle est très utilisée dans le monde des logiciels libres et même commerciaux. Elle est aussi utilisée en interne de l’environnement de développement Code::Blocks, alternative gratuite de la suite Visual de Microsoft. Code::Blocks est maintenant disponible sur de nombreux environnements, dont Windows, Linux, MacOS et installé depuis peu, en standard, sur les machines Linux à l’Onera. L’avantage de Code::Blocks par rapport à Eclipse, logiciel de développement gratuit également, est qu’il intègre un outil de développement d’interface graphique, wxSmith, qui a été utilisé ici pour développer l’interface d’Ovni.

Une première maquette de faisabilité a été développée fin 2014 à partir d’un exemple de démonstration intégré à la distribution de wxWidgets (« penguin ») et permettant de mouvoir à l’aide de la souris un objet 3D dans une fenêtre OpenGL. Cette maquette a permis notamment des tester la construction d’interface via wxSmith et d’intégrer quelques premiers éléments repris d’Ovni.

A partir d’Avril 2015, la réécriture d’Ovni a été confiée à un stagiaire d’une école d’ingénieur, l’ECE Paris, sur une durée de 3 mois. L’objectif premier du stage était la réécriture de la partie interface graphique. Toutefois, wxWidgets s’interface plutôt avec C++ et le stagiaire s’est vite tourné vers une réécriture plus profonde du code pour pouvoir d’avantage bénéficier des possibilités de C++. Ce faisant, le travail à réaliser étant très important, le travail était loin d’être terminé en fin de stage. Un autre stage a été proposé pour 2016, mais n’a pas été pourvu faute de candidat adéquat. Ce n’est donc qu’à partir du début 2018 que le développement a été repris (Home work de Gérard DURAND en retraite !) : corrections de bugs laissé par le stagiaire, améliorations diverses et transposition de fonctions de la version 4 d’Ovni vers la version 5.

La version 4 d’Ovni était, sous Windows, une application 32 bits. La réécriture et l’adoption de wxWidgets (actuellement en version 3.2.0) a permis d’en faire une application 64 bits. Autre point très intéressant : il est maintenant possible d’utiliser le débuggeur gdb, intégré dans Code::Blocks pour traquer les erreurs du code : avec Tcl/Tk, ce n’était pas possible car, Tcl/Tk ayant la main sur le code, seul le débogage à « l’ancienne », via des printf disséminés un peu partout était faisable !

Ovni est actuellement compilé avec la version 8.1 de gcc/g++ alors qu’on se contentait de la version 4.9 dans les dernières versions avec Tcl/Tk.

D’un point de vue pratique, on s’est attaché à ce que l’apparence de l’interface graphique réalisée avec wxWidgets reste proche de celle de la version précédente, sous Tcl/Tk. Ainsi, la documentation de la V5 est souvent très proche de la V4. Quelques différences de comportement et fonctionnalités apparaissent toutefois, motivées par l’usage, et demanderont éventuellement un petit temps d’adaptation à l’utilisateur.

# domaine d’application

## Historique du développement.

Le logiciel de Calcul du Rayonnement InfraRouge des Avions, développé à l’Onera depuis plusieurs années, utilise des bases de données géométriques 3D d’avions.

Historiquement, la géométrie 3D des avions a d’abord été introduite dans Crira par une interface avec le logiciel de CAO Euclid, puis a migré vers Infovision, compatible d’Euclid. L’un des inconvénients de cette approche est qu’un utilisateur de Crira devait posséder un de ces progiciels, lourds et coûteux. De plus, dans l’utilisation qui en était faite dans Crira, il n’y avait que très peu de fonctionnalités de ces logiciels qui étaient utilisées. De plus, l’avion était reconstruit à chaque exécution de Crira, ce qui n’était pas optimal. Suite à un rapprochement avec Oktal-SE, l’utilisation du format de base de données SDM, format propriétaire d’Oktal-SE, a été introduite dans Crira. C’est un format assez répandu au sein des utilisations de géométrie 3D liés à la DGA. Toutefois, la lecture et l’exploitation des fichiers 3D d’avions étaient toujours gérées par une interface avec des outils d’Oktal, imposant à l’utilisateur l’achat d’une licence. Dans la version 6 de Crira nous avons introduit des outils de manipulation et de gestion des objets 3D, en partie issus des études Matisse, ce qui a permis d’obtenir une version de Crira totalement autonome, c'est-à-dire ne nécessitant plus l’acquisition d’outils externes par l’utilisateur. Afin de se dégager au maximum des outils propriétaires d’Oktal, il nous fallait également avoir un logiciel capable de lire les fichiers au format SDM pour d’abord les visualiser, mais également pour y apporter un certain nombre de modifications et d’adaptations, nécessaires à une utilisation par Crira. C’est la raison du développement d’OVNI.

Le développement a commencé sous la forme d’un projet tuteuré par des élèves de l’IUT informatique d’Arles en 2004. Le but de ce stage était surtout de bâtir un projet de logiciel en identifiant les besoins des utilisateurs et de proposer des solutions logicielles adaptées [DR1]. A l’issue du stage, une préversion de logiciel a été fournie. Le cœur du logiciel est développé en langage C, s’appuie sur un rendu OpenGL des objets 3D et utilise une bibliothèque logicielle Glut. C’est lors d’un stage de l’année suivante qu’ont été réellement développées les premières versions. La seule bibliothèque Glut ne permettant pas de gérer facilement une interface avec les boutons, des menus, des fenêtres d’entrée/sortie, c’est dans cette phase qu’a été introduite d’abord GTK, interface très utilisée dans le monde Unix. Mais, suite à des difficultés d’implantation et de compatibilité avec l’existant Onera, c’est ensuite vers Tcl/Tk que le choix s’est porté. Tcl/Tk est toujours utilisé actuellement. C’est un langage de scripts permettant de créer et gérer facilement des éléments d’interface graphique comme des boutons, des cases à cocher, des menus déroulants, etc.… Il possède en natif de nombreuses fonctions directement exploitables. Pour celles nécessitant des développements spécifiques, une interface vers des fonctions écrites en langage C est disponible. A l’issue de ce second stage [DR2], nous disposions d’une version d’OVNI possédant déjà certaines des fonctionnalités définies dans le projet et une interface préfigurant celle disponible actuellement. Le projet avait été développé sur Linux à l’IUT et fonctionnait sur Sun Solaris à l’Onera. Certaines des fonctions développées pour OVNI sont réutilisées dans Crira.

Suite à ce stage, le développement a été repris à Palaiseau afin d’implanter des fonctionnalités utiles mais encore manquantes. Certaines nouvelles fonctions, non prévues à l’origine, ont été introduites comme par exemple l’importation d’autres formats 3D. De plus, un portage vers Windows a été réalisé à ce moment.

Un troisième stage, effectué en 2006, est reparti des développements introduits à Palaiseau et étendu les fonctionnalités [DR3].

Depuis ce stage, le développement d’OVNI a de nouveau été repris à Palaiseau. Il a tout d’abord fallu corriger un certain nombre de bugs restés à l’issue de ce troisième stage, bugs qui causaient des plantages aléatoires sous Windows, mais sans incidence sous Sun Solaris apparemment. Au fur et à mesure des utilisations, notamment dans le cadre d’un PEA (Gabarits), de nouvelles fonctionnalités ont été introduites.

Le format de base des fichiers 3D utilisés dans Crira est toujours le format SDM d’Oktal. C’est la raison pour laquelle c’est ce format qui est utilisé comme base dans OVNI en lecture et en écriture. Il est envisagé d’utiliser dans Crira un autre format, issu des travaux du groupe 3D interne à l’Onera/Dota (g3d). OVNI pourrait facilement migrer vers ce format, mais un outil équivalent est en cours de développement pour g3d. Actuellement, OVNI est en avance sur cet outil g3d, mais ce ne sera pas toujours le cas. Tout dépend de l’avenir de ce groupe de travail et des fonctionnalités supportées par le visualiseur g3d.

Jusqu’à la version 3.31 d’Ovni, on utilisait un programme mktclapp permettant de mélanger du code en C et des scripts en Tcl/Tk. Mktclapp est assez ancien et n’est plus développé par son auteur. Mktclapp est fourni sous forme d’exécutable pour Windows, Linux/Redhat/CentOS (et éventuellement Solaris). Le code source en C de mktclapp est fourni avec les sources d’Ovni. Il est utilisable jusqu’aux versions 8.4 de Tcl/Tk. Certains de des appels à Tcl/Tk générés par mktclapp ne sont plus supportés depuis les versions Tcl/Tk 8.5. On construit alors Ovni sous la forme d’une librairie dynamique (dll sous Windows, fichier .so sous Linux) appelable depuis un code Tcl/Tk. Cette librairie peut être utilisée comme telle ou intégrée dans un exécutable autonome généré par Tclkit.

Comme décrit dans le préambule du paragraphe précédent, il est devenu évident que le logiciel ne pourrait plus être développé efficacement en raison de l’obsolescence de certains logiciels/bibliothèques intermédiaires. C’est pourquoi, il a été décidé de changer de bibliothèque graphique et de passer à wxWidgets. D’autres choix auraient été possibles, mais c’est notamment parce que Code::Blocks, interface de développement utilisée par l’auteur, est bâti autour de cette librairie et intègre des outils de développement graphique comme wxSmith, que ce choix a été favorisé.

## Fonctionnalités principales du code

Les fonctionnalités principales du code qui seront détaillées dans ce document sont les suivantes :

* Lecture de fichiers 3D au format SDM d’Oktal-SE (fichiers .bdd) ;
* Affichage dans une fenêtre OpenGL des objets lus : objet plein, avec ou sans les points des sommets, avec ou sans les arêtes des facettes ;
* Manipulation des objets 3D : rotations, translations, zoom avec la souris ou à l’aide de touche du clavier ;
* Repérage d’éléments particuliers comme des points, des facettes, des objets : sélection manuelle par menu ou par clic de souris ;
* Ajout de formes génériques simples : sphère ou ellipsoïde, cylindre, cube ou parallélépipède, cône, facette isolée, icosaèdre (nouveauté de la V5 qui peut permettre, par itération, de générer un autre type de sphère) ;
* Ajout de points, de facettes : fonctions utiles pour corriger des bases de données avec des erreurs, des trous ;
* Modifications de coordonnées de points (correction de bdd) ;
* Rotations globales, symétries globales, changement de signes sur certains des axes des bases de données, permutation d’axes ;
* Identification ou entrée de numéro de groupe (fonction indispensable pour Crira) ;
* Identification ou entrée de numéro de matériau
* Modification et stockage de la palette de couleurs à affecter aux groupes et/ou aux matériaux ;
* Identification du sens des normales (fonction indispensable pour vérifier si elles sont correctement orientées en vue des calculs de réflexion sur les différentes parties d’avions) ;
* Modification du sens des normales ;
* Modification du sens de parcours des facettes (fonction proche de la précédente mais qui peut maintenant être distincte) ;
* Recalcul des normales ;
* Stockage d’une capture d’écran dans un fichier graphique
* Masquage temporaire d’objets, suppression d’objets ;
* Masquage temporaire de facettes, suppression de facettes ou de points ;
* Création d’objets par symétrie ou translation d’objets existants ;
* Modification d’échelle de la base de donnée, translation de la base de donnée (fonctions utiles pour convertir des millimètres en mètres, des pouces en mètres, déplacer l’origine par exemple) ;
* Importation de formats de fichiers autres : .obj, .3ds, .ply, .m3d, .g3d, .off, .stl ;
* Fusion de bases de données, éventuellement de formats différents ;
* Enregistrement des fichiers : format SDM Oktal .bdd (par défaut), possibilité d’enregistrer dans les formats .obj, .off, .g3d, .stl ;
* Aide en ligne.

OVNI ne remplace pas un modeleur, mais possède certaines des fonctions de ce type de logiciels, du moins celles permettant de modifier un avion existant et l’adapter à un usage dans Crira, notamment par l’ajout des numéros de groupes. Cette notion de groupe, voisine mais différente de celle de matériaux, permet de regrouper des facettes d’avion qui ont un comportement optique et/ou thermique identiques. Par exemple, on distingue 3 groupes sur une aile d’avion : les bords d’attaque (partie avant), l’intrados ou l’extrados de l’aile (partie médiane) et les bords de fuite (partie arrière, typiquement les ailerons). Les peintures et donc les matériaux de surface peuvent être les mêmes, mais le comportement thermique différent du fait de la position de la facette par rapport à l’écoulement de l’air : un bord d’attaque peut être légèrement plus chaud que la partie médiane de l’aile, elle-même plus chaude qu’un bord de fuite. Notons que la notion de groupe existe en natif dans le format SDM, existe aussi dans le format g3d, mais est absente de tous les autres formats connus et/ou supportés.

Le format SDM est un format basé sur l’utilisation de facettes. Les facettes sont regroupées au sein d’objets. Une base de données peut ne comporter qu’un seul objet. Chaque facette est composée d’un certain nombre de sommets repérés par un numéro de points. Le nombre de sommets d’une facette peut être quelconque. Le tableau des points utilisés par chaque objet comporte les coordonnées en X, Y et Z de chacun d’eux. La position correspond au numéro du point utilisé dans les facettes. En version Tcl/tk d’Ovni, l’indice 0 servait à stocker éventuellement quelques informations. Ainsi, numéro de points et indice dans le tableau concordaient dans tous les cas. Cette façon de faire n’a pas été reconduite lors de la réécriture en C++. Une des raisons à cela, est qu’on utilise la classe de vecteur en C++ et cette classe comporte en natif des outils bien plus complets et évolués que ce qu’on pouvait faire à l’aide de l’indice 0 en C. Du coup, on a maintenant un décalage d’une unité entre l’indice et le numéro d’un point, d’une facette (gare aux bugs !). Une facette comporte au minimum 3 points, mais peut donc en comporter davantage. Une facette à 3 points est forcément plane. Dans ce cas, la normale au barycentre de la facette est bien définie. Une facette à plus de 3 sommets peut être plane, mais peut aussi être gauche. La normale au barycentre peut alors ne pas être bien définie, surtout si on la calcule à partir de 3 sommets seulement. Dans Crira, on utilisera uniquement des facettes à 3 sommets, bien que ce ne soit pas une contrainte de ce format SDM ni même une contrainte d’OpenGL. Par contre, cela le devient pour le moteur de rendu interne de Crira.

Pour les calculs de rendu, les calculs de réflexion sur les facettes, Crira a besoin de connaître les normales au barycentre de chaque facette, mais également les normales aux sommets des facettes. La normale au barycentre permet de connaître l’orientation générale d’une facette. Les normales aux sommets sont utilisées par le lissage de Gouraud qui, par interpolation à partir de la connaissance de ces normales, recrée une apparence de surface continue en éliminant les discontinuités entre facettes. C’est une approximation qui permet de traiter assez correctement la visualisation de surfaces lisses de l’objet en passant par une décomposition des objets en facettes planes sans avoir besoin d’une modélisation en 3D des surfaces, utilisant par exemple des surfaces de Bézier, comme ce qui est fait avec d’autres formats de fichiers (par exemple les formats utilisés dans Catia, logiciel Dassault).

La plupart des formats de bases de données permettent de donner une couleur à des facettes ou encore d’y attacher des textures. Ces fonctionnalités n’ont pas été implémentées dans OVNI car ces informations ne sont pas utilisées par Crira : les propriétés infrarouge n’ont pas souvent de rapport direct avec les couleurs et les textures. La colorisation de facettes ne sera faite dans OVNI que pour caractériser les différents groupes (au sens de Crira), matériaux, objets ou facettes.

OVNI supporte en entrée un certain nombre de formats de bases de données d’objets. Seul un sous-ensemble des diverses possibilités de chaque format est supporté. Ainsi, il se peut que sur certaines bases de données le résultat ne soit pas celui escompté : objet manquant, mal placé, ... OVNI a été testé sur près de 3000 bases de données différentes, mais même ainsi, toutes les fonctionnalités de tous les formats n’ont pas été explorées. Au fur et à mesure de l’usage, les fonctions manquantes seront ajoutées dans OVNI.

# documents de rÉfÉrence

*Citer une seule fois leur référence ci-dessous. Les appeler dans le texte ensuite seulement par leur numéro dans la liste :*

**Documents de référence :**

Ce sont les documents qui sont utilisables comme support, bibliographie, etc. Ils sont repérés dans le texte par leur numéro dans la liste.

|  |  |
| --- | --- |
| 1. Visualisation 3D/2D Onera. Cahier des charges | Projet Tuteuré.  LP\_CDC\_Onera\_v3.0.pdf (Décembre 2004) |
| 1. Réalisation d’un outil de visualisation d’objets 3D et mise au point de fonctions de calculs. | L. Lagouarde  Rapport de stage (Juin 2005) |
| 1. Ajout de fonctionnalités à un outil de visualisation d’objets 3D | R. Sourbé  Rapport de stage (Juin 2006) |
| 1. Format SDM 3.0 – OKTAL R&D | G. Giannesini  Format SDM V3.0.pdf  92 0MC 007 SPL vers 03 AA (Août 1995) |
| 1. Rapport de synthèse de l’activité 2008 du groupe de travail G3D | P. Barillot, L. Labarre, K. Souadia  RT 3/13532 DOTA – Avril 2009 |
| 1. Rapport d’activité 2009 du groupe de travail G3D | P. Barillot, L. Labarre, J. Pierro  RT 2/14504 DOTA – Août 2010 |
| 1. Rapport de Stage | Jordy Dias. ECE.  Cycle ingénieur 2ème année 2014/2015 |

# dÉfinitions et abrÉviations

|  |  |
| --- | --- |
| BDD | **B**ase **D**e **D**onnées |
| CESAME | **C**entre **É**tatique de **S**imulation **A**vancée de **M**odélisation **É**lectromagnétique |
| CRIRA | **C**ode de calcul du **R**ayonnement **I**nfra**R**ouge des **A**vions |
| DA | **D**assault **A**viation |
| DGA | **Direction Générale de l’Armement** |
| DRi | **D**ocument **R**éférence n°i |
| MU | **M**anuel **U**tilisateur |
| PEA | **P**lan d’**É**tude **A**mont |
| SIR | **S**ignature **I**nfra**R**ouge |

# Interface du logiciel OVNI

OVNI se présente sous la forme d’un exécutable classique. Il nécessite la présence d’une librairie dynamique compilée wxWidgets (3.2.0 actuellement). Bien que ce ne soit pas indispensable, il est préférable que le code et la librairie aient été compilés avec la même version de compilateur (ici gcc/gcc 8.1). D’après quelques informations vues sur Internet, la non concordance est surtout liée au langage C++ et peut conduire à des soucis à l’exécution en cas de non-respect de cette règle (mais ça peut aussi fonctionner !).

*Note* : il est possible de générer Ovni sous la forme d’un exécutable statique, sans aucune librairie dynamique annexe. L’exécutable est alors assez gros, mais totalement autonome.

En version 4, sous Tcl/Tk, il y avait aussi une version Linux. Cette possibilité existe toujours en version 5, mais faute de disponibilité d’une machine adéquate, elle n’a pas été testée. L’adaptation ne devrait pas poser de gros problèmes.

## Fenêtre principale

Sous Windows, lorsqu’on lance OVNI en double cliquant sur un fichier dont l’extension est .bdd (à condition d’avoir associé cette extension avec OVNI), on obtient une fenêtre comme celle ci-dessous.

L’avion est ici un Mirage 2000, vu de dessus, dont la base de données est compatible du système d’axes de Crira : l’axe des X est horizontal (en rouge) et pointe vers la droite, l’axe des Y (en vert mais non visible ici) est aussi horizontal, pointe vers le haut de l’image et est en gros dans l’axe avant-arrière de l’avion. L’axe des Z (en bleu) est vertical et pointe ici vers l’utilisateur.

À l’ouverture d’un fichier, le cadrage est automatiquement adapté à la taille de l’objet en base de données. La fenêtre totale contenant l’objet est par défaut de 816\*647 pixels, la partie OpenGL étant alors de 800\*400 pixels (la taille réelle des images est inférieure de 4 pixels, soit 796\*396). On peut redimensionner la fenêtre jusqu’à la taille maximale écran par les moyens classiques à la souris ou encore via le bouton Plein écran standard.



La première ligne en haut de l’interface est une ligne de menus.

La seconde ligne comporte divers boutons permettant d’agir sur la visualisation.

La troisième partie est la fenêtre OpenGL, présentant l’avion sur un fond bleu. Les petits traits en rouge en haut, en bas, à droite et à gauche délimitent la trace de la boîte englobante de l’avion. Cette boîte est délimitée par les minima et maxima des coordonnées dans chacun des trois axes. Des mouvements à la souris, effectués dans cette fenêtre, permettent d’effectuer des rotations de l’avion, des déplacements latéraux, des zooms.

La quatrième partie, en bas de l’écran, présente trois curseurs (ou sliders en anglais ou encore parfois dénommés ascenseurs) permettant de faire pivoter les axes de visualisation de la base de donnée. On utilisera par la suite le mot curseur plutôt qu’ascenseur ou slider. L’amplitude est de 360° sur chacun des 3 axes. C’est redondant, mais volontaire, car plusieurs triplets de valeurs donnent la même présentation.

### Mouvements de base à la souris

Ces mouvements sont obtenus par déplacements de la souris dans la fenêtre OpenGL de base, sans autre fenêtre de menu ou de configuration particulière.

**Bouton gauche maintenu enfoncé** : permet d’effectuer des rotations de l’objet de droite à gauche et de bas en haut. Deux modes de rotation sont possibles (configurables via le menu Fichier/Préférences) : le mode direct ou le mode trackball :

* Le mode direct est celui introduit à l’origine dans OVNI : les déplacements en X et Y de la souris affectent directement les rotations sur les axes X et Z de la base de données. Seuls les curseurs en X et Z en bas de l’écran bougent. Dans ce mode, une rotation directe à la souris autour de Y n’est pas possible ainsi. C’est la raison pour laquelle les curseurs en bas de l’écran ont été introduits.
* Le mode trackball a été introduit en 2009. Ce mode existait de base dans l’application de démonstration de wxWidgets « penguin ». Il simule l’utilisation d’une boule. En fait, les mouvements de la souris sont considérés comme pilotant les axes X et Y de l’écran au moment où le déplacement débute. On peut donc agir sur les 3 axes de la base de données simultanément, OVNI se chargeant de faire le changement d’axes écran -> base de donnée. Les curseurs du bas restent affichés en angles de la base de données.

**Bouton droit maintenu enfoncé** : permet de zoomer. Un mouvement de haut en bas agrandi l’objet, un mouvement de bas en haut rétréci l’objet (*Note : c’était l’inverse dans la version Tcl/Tk*). Si de plus, on appuie sur la touche Maj. à gauche du clavier, on obtient un zoom plus rapide. Lorsque l’objet devient gros, une fonction automatique réduit l’incrément du Zoom.

**Rotation de la molette** : permet de zoomer/dézoomer.

### Mouvements de base au clavier

Ces mouvements sont obtenus en appuyant sur certaines touches du clavier lorsque la fenêtre OpenGL de base est la fenêtre active.

* **Touche +** : Agrandi le zoom. Fonctionne avec la touche + en haut du clavier ou la touche + du pavé numérique.
* **Touche -** : Diminue le zoom. Fonctionne avec la touche - en haut du clavier ou la touche - du pavé numérique.
* **Touche Flèche droite** : Déplace l’objet vers la droite.
* **Touche Flèche gauche** : Déplace l’objet vers la gauche.
* **Touche Flèche bas** : Déplace l’objet vers le bas.
* **Touche Flèche haut** : Déplace l’objet vers le haut.
* **Touche Flèche droite + touche Maj** (du pavé des 4 touches fléchées) : Rotation de l’objet autour de l’axe Z (axe dessiné en bleu).
* **Touche Flèche gauche + touche Maj** : Idem mais rotation inversée.
* **Touche Flèche bas + touche Maj** : Rotation de l’objet autour de l’axe X (axe dessiné en rouge).
* **Touche Flèche haut + touche Maj** : Idem, mais rotation inverse.
* Les quatre rotations sont également possibles via les touches 6, 4, 2 et 8 du pavé numérique si la touche de Verrouillage Numérique (Verr. Num.) est désactivée.
* **Touche Page Up** : Agrandi le zoom rapidement (⬄ - et Touche Maj.).
* **Touche Page Down** : Diminue le zoom rapidement (⬄ + et Touche Maj.).

Si les touches sont maintenues enfoncées, la fonction de répétition de touche permet d’obtenir une action continue.

### Les menus

La barre de menus contient plusieurs menus, chacun se déclinant en un certain nombre de sous-menus.

Le menu Fichier permet d’accéder notamment à l’ouverture de fichiers, à l’enregistrement de fichiers, aux Préférences où on va pouvoir paramétrer certaines fonctions d’OVNI, à l’affichage des propriétés de la base de données en cours ou aux paramètres de la carte vidéo.

Le menu Affichage permet de choisir entre différents modes d’affichages ou diverses vues prédéfinies.

Le menu Primitive permet d’ajouter une forme simple comme une sphère, un cube, un cône,…

Le menu Repérage permet d’afficher des sous-menus pour identifier des points, des facettes, des objets, des groupes de facettes, des matériaux par leurs numéros. C’est aussi via ce menu qu’on va pouvoir repérer les facettes dont les normales sont mal orientées.

Le menu Image s’utile pour enregistrer la fenêtre OpenGL dans 3 formats disponibles, jpeg, png ou ppm.

Le menu Outils comporte un certain nombre d’outils qui vont permettre de manipuler la base de données et d’identifier des objets, des facettes, des points par des clics à la souris.

Le menu Transformations agit sur toute la bdd pour la déplacer, en faire une rotation. Contrairement aux fonctions flèches et à la souris, qui n’agissent que sur l’écran, ici ce sont les coordonnées dans la base de données qui seront modifiées.

Le menu Options permet de paramétrer certaines fonctions de colorisation, relire ou stocker une palette de couleurs.

Le menu Aide affiche dans une fenêtre un aide rapide des diverses fonctions accessibles au clavier et à la souris.

#### Le menu Fichier

|  |
| --- |
|  |

Le menu Fichier comporte plusieurs sous-menus.

Le premier, Ouvrir, est la fonction classique d’ouverture de fichier. Une boîte de dialogue permettant de choisir quel est le fichier à ouvrir s’ouvre dans l’espace de travail. Par défaut, tous les formats reconnus par OVNI sont proposés. On peut ne choisir qu’un seul des types, ou tous les types de fichiers (\*.\*), y compris ceux dont les extensions ne sont pas connues. C’est l’extension qui détermine le type de fichier à lire. En version Tcl/Tk, le contenu du fichier était analysé pour décider du type de fichier à ouvrir. Cette possibilité n’a pas été reprise en version wxWidgets.

Le second sous-menu est analogue à Ouvrir, mais il ne sert qu’à ré-ouvrir le fichier courant. Il n’y aura pas de boîte de dialogue d’ouverture de fichier. Utile si on change une option de visualisation ou de lecture du fichier et qu’on veut voir l’effet sur le fichier original. Toutefois, si la bdd a été modifiée, OVNI propose d’abord de la sauvegarder pour ne pas perdre le travail effectué. Si on accepte, on pourra l’enregistrer sous un nouveau nom, mais c’est ce fichier qui sera ré-ouvert.

Le troisième sous-menu est analogue à Ouvrir, mais on l’utilise pour ouvrir un autre fichier et l’ajouter à la bdd existante. S’utilise par exemple si on possède un fichier ne contenant qu’un bidon et qu’on veuille ajouter ce bidon à un avion qui n’en possède pas. On verra par la suite comment déplacer et ajuster ce bidon. Il existe aussi une fonction de fusion d’objets sélectionnés accessible via le menu ou l’icône « Sélection et déplacements ».

Le quatrième sous-menu est la fonction classique Enregistrer. Elle sauvegarde, par défaut, la base de données au format au format SDM Oktal (extension .bdd). ***Attention***, dans le cas où on a ajouté d’autres bases de données, c’est le premier nom entré qui est retenu dans cette fonction. Si on veut conserver séparément chacune des bdd et notamment la première, il faut alors passer par le menu suivant afin de pouvoir l’enregistrer sous un autre nom.

Le cinquième sous-menu est une fonction permettant d’enregistrer la bdd sous un autre nom que celui ayant été utilisé dans Ouvrir. Par défaut, c’est au format .bdd qu’elle est enregistrée. La possibilité d’enregistrer dans le format .off a pas été reprise dans cette version wxWidgets (en fait, très rarement utilisée dans le contexte Crira). Il est maintenant possible d’enregistrer sous les formats Wavefont obj (\*.obj), XML G3D (\*.g3d), Objet File Format (\*.off) ou Stéréolithographique (.stl) dans ses 2 variantes Asii et binaire. À noter que les formats autres que .bdd sont peu utilisés dans un contexte général Crira car limités du point de vue des attributs stockables (pas de normales, pas de groupes,…).

|  |
| --- |
|  |

Le sous-menu « Propriétés » permet d’afficher différentes propriétés de la base de données. On y trouve notamment, le nombre total d’objets, le nombre total de points, le nombre total de facettes, le nombre d’arêtes, le nombre de groupes et de matériaux comme le montre l’exemple ci-contre. La décomposition des facettes en fonction du nombre de points qui les composent est également affichée. Dans la partie haute on trouve le nom de la base de données (nom du fichier .bdd). En cas de fusion, les différents noms sont ajoutés les uns aux autres et séparés par le signe +. Dans la partie basse sont affichées les valeurs min et max sur les 3 axes ainsi que les dimensions de la boîte englobante, informations utiles pour déterminer dans quelles unités sont fournies les coordonnées de la base de données. Le bouton « OK » permet de fermer cette fenêtre informative.

|  |
| --- |
|  |

Le sous-menu « Préférences » permet d’afficher différentes options d’utilisation d’OVNI. La plupart des choix proposés sont sauvegardés dans un fichier « Ovni.ini » situé dans le même répertoire que l’exécutable.

La fenêtre de ce menu à l’aspect ci-contre. Les deux valeurs « Longueurs des axes » et « longueur des normales » permettent d’ajuster ces longueurs, représentées par des traits en couleur dans la fenêtre OpenGL et sur les images. Les valeurs entrées correspondent à un coefficient de proportionnalité par rapport à la diagonale de la boîte englobante. Ce coefficient est compris entre 0 et 1. À noter que cette valeur est utilisée telle que lorsque le Zoom est de 1. Sinon, la valeur réellement utilisée est ajustée en fonction du Zoom afin que les normales ou les axes aient toujours une longueur raisonnable par rapport à la fenêtre de tracé. Le rayon de la source est utilisé pour tracer un petit cercle rempli de jaune et symbolisant la position de la source d’éclairage utilisée par OpenGL. L’affichage de cette source est optionnel (voir le sous-menu Affichage).

La case à cocher « Anti-aliasing géré par OpenGL » est prévue pour laisser le soin à OpenGL de traiter l’anti-aliasing plutôt que de le faire par le hardware de la carte graphique. Cette fonction a de l’intérêt pour lisser le tracé des arêtes lors du tracer en mode filaire. Le rendu est bien plus joli qu’en mode hardware. Elle active aussi l’utilisation du mot clé GL\_MULTISAMPLE dans OpenGL, ce qui permet de gérer via OpenGL l’anti-aliasing des polygones (ou bords de facettes). Toutefois, ceci ne fonctionne que si le driver de la carte graphique est configuré pour supporter cette option. Il existe d’autres façons d’obtenir de l’anti-aliasing des bords de polygones mais ceux-ci sont soit compliqués à mettre en œuvre soit peu performants car entièrement software. Cette option n’est pas cochée par défaut car elle peut ralentir l’affichage de façon importante si la base de données comporte beaucoup de facettes et d’arêtes.

La case à cocher « Forcer la lecture des fichiers .obj dans 1 seul Objet 3D » n’est pas activée par défaut. Le format Wavefront obj permet de différentier plusieurs objets. Toutefois, la numérotation des points en interne du fichier est cumulative et ne sépare pas de fait les objets. Cette option permet donc de ne plus différentier les différents objets et ainsi, en utilisant directement la numérotation des points, faire en sorte qu’Ovni considère que la base de données en question ne possède qu’un seul objet. S’avère utile sur certains fichiers .obj un peu tortueux.

La case à cocher « Lecture optimisée des tableaux de points des fichiers de type .obj » est désactivée par défaut (dans la version actuelle). Pour ce format de fichiers, il n’y a qu’un seul tableau de points commun à tous les objets. Dans les autres formats de fichiers, on a autant de tableaux de points que de tableaux d’objets. La première solution adoptée dans OVNI était de lire l’ensemble des points, de dupliquer ce tableau dans tous les objets présents puis de faire un tri pour ne garder que les points utilisés par chacun des objets. C’est une solution simple et rapide mais qui présente l’inconvénient, sur de grosses bases de données, d’occuper temporairement de beaucoup d’espace mémoire. Dans certains cas, en particulier sous Windows, on dépassait la capacité d’adressage des 32 bits de Windows. C’est la raison qui a amené à écrire une procédure spécifique de lecture optimisée, recherchant dès la lecture des points, dans quels objets ils sont utilisés, et donc à les ranger directement là où ils sont utilisés. Cette procédure existait déjà dans les dernières versions Tcl/Tk. Toutefois sa réécriture en version wxWidgets s’est avérée plus délicate, a toujours besoin, au moins temporairement, de gros tableaux de sommets et vecteurs. C’est la raison pour laquelle elle est optionnelle. De plus, elle allonge sensiblement le temps de lecture du fichier. On peut donc la désactiver si besoin. Le souci lié à la capacité d’adressage de la mémoire nécessaire n’est plus un problème en compilation 64 bits.

*Note : On obtiendra le même effet, et même encore mieux, en utilisant après coup la Simplification de Base de Données.*

La case à cocher « Lecture du décalage éventuel des objets dans les fichiers .3ds » permet de récupérer une information de décalage entre objets incluse dans les fichiers Autodesk 3ds (via une matrice dépendant des « nodes » et/ou des « meshes »). Actuellement cette case n’est pas cochée par défaut car le traitement de l’information récupérée ne donne pas toujours les bons résultats. Normalement la détection du décalage devrait être automatique, comme dans certains logiciels de démonstration de lecture 3ds (comme 3dsplay ou player fournis avec les versions de la librairie lib3ds). Le problème est maintenant corrigé dans la plupart des cas, mais il reste quelques cas rares de fichiers où cela ne fonctionne pas correctement. Une option spécifique du menu Fichier s’ajoute automatiquement pour proposer l’autre solution de lecture de fichiers 3ds (avec ou sans prise en compte du décalage). On peut alors tenter de le relire avec cette autre option.

Le groupe de données suivant concerne le calcul et la visualisation des normales aux sommets des facettes. Si la première case à cocher est désactivée, on utilise les normales telles qu’elles sont fournies dans la base de données, ou calculées à la lecture du fichier en cas d’absence. Ces normales sont utilisées pour le lissage de Gouraud (on parle également d’ombrage de Gouraud). Si la case est cochée, on va pouvoir effectuer des tests en temps réel en ajustant des valeurs angulaires de seuillage. L’explication détaillée de ces valeurs sera fournie plus loin, mais pour l’instant nous dirons rapidement qu’on compare chaque normale aux sommets d’une facette avec la normale au barycentre de la facette. Si l’écart angulaire est supérieur au seuil (première valeur entrée, par défaut à 35°) alors il faut considérer qu’en ce sommet la facette est en fait anguleuse. La normale au sommet calculée à l’aide des normales aux barycentres de toutes les facettes utilisant ce sommet ne doit pas être utilisée. Il n’y a donc pas de lissage pour ce sommet. Toutefois, il y a des cas plus complexes où sur 4 facettes adjacentes (ou plus) utilisant un même sommet, certains peuvent être lissées entre elles et d’autres pas. C’est la raison de la présence du deuxième angle de seuillage qui est ici entré comme un facteur multiplicatif par rapport au premier angle entré, par défaut 1.6. L’action des boutons fléchés est réalisée en temps réel sur la visualisation (*la base de données n’est pas modifiée à ce niveau*). On peut donc modifier ces valeurs sans forcément trop comprendre leur fonction et observer l’effet sur la visualisation de l’objet en 3D.

La case à cocher suivante peut être utilisée pour calculer dès la lecture du fichier les normales aux sommets si celles-ci sont absentes du fichier. Au cas où le fichier contient déjà ce type de normales, Ovni demande la confirmation du calcul à l’utilisateur. Cette case est cochée par défaut. Néanmoins, ce calcul peut être long sur de grosses bases de données et il peut être utile de ne pas le faire d’emblée, surtout si on n’utilise pas le lissage de Gouraud. On peut gagner pas mal de temps. Le calcul pourra se faire lors d’une seconde lecture ou en utilisant une option de calcul des normales que l’on verra ailleurs.

La case à cocher suivante permet d’indiquer si on désire stocker lors de l’enregistrement les valeurs des normales aux sommets telles qu’elles ont été utilisées et éventuellement modifiées en temps réel par l’action des boutons fléchés vus ci-dessus. Par défaut cette case n’est pas cochée afin d’éviter de modifier le fichier par inadvertance. Mais si on désire réellement utiliser les valeurs des normales telles qu’elles sont vues dans OVNI dans le logiciel CRIRA, en particulier pour traiter correctement les problèmes de reflets, il faut la cocher, mais n’enregistrer le résultat qu’une fois trouvé le bon paramétrage des valeurs angulaires des seuils (voir en annexe le calcul des normales dans OVNI).

La case à cocher suivante permet de forcer à ce que toutes les facettes soient considérées comme non planes lors d’un recalcul de toutes les normales. En effet, lorsque certaines facettes sont déclarées planes et enregistrées comme telles dans un fichier bdd (les normales aux sommets d’une facette sont égales à la normale au barycentre), par défaut cet attribut est conservé. Dans certains cas, cela peut-être gênant et il vaut mieux tout recalculer, y compris les normales aux sommets, puis re-déclarer explicitement les facettes qui doivent rester planes.

La case à cocher suivante permet de lancer ou non la détection des doublons d’arêtes dans les facettes. Par défaut, cette case est activée, mais cette détection n’est véritablement utile que si on désire rajouter des points dans des facettes, subdiviser des arêtes en 2 ou plus de segments ou encore souder des points. Si ce n’est pas le cas, on peut tout à fait décocher cette case car l’opération de détection des doublons peut être longue à exécuter sur de grosses bases de données. De toute façon, une limite est imposée dans le code sur le nombre de facettes afin de ne pas lancer cette détection s’il y en a vraiment beaucoup. Cependant, la touche clavier « f » (pour forcer) permet quand même de forcer cette détection, mis dans ce cas, il faut s’attendre à des traitements très longs.

Le menu suivant permet de paramétrer le temps entre deux sauvegardes automatiques de la base de données. Par défaut ce temps est de 5 minutes. La sauvegarde automatique crée un fichier Autosave.bdd soit dans le répertoire de base de l’application, soit dans le répertoire ou se trouve la base de données d’origine. Au cas où ce répertoire n’est accessible qu’en lecture, car la bdd est sur un cd-rom ou sur un lecteur réseau où on n’a pas les droits en écriture, c’est le premier choix de répertoire qui sera utilisé. Ces sauvegardes sont utiles en cas de mauvaises manip en cours de travail ou encore en cas le plantage de l’application. Cela arrivait parfois sous Windows en version Tcl/Tk, n’est pas encore arrivé en versions wxWidgets, mais au cas où cela arriverait, certainement à cause de bugs soit directement impliquables au code source d’OVNI qu’on trouvera tôt ou tard, soit de bugs dans les librairies tierces sur lesquelles nous n’avons que peu ou pas d’emprise, permettrait de ne pas perdre un travail de mise en forme parfois long et délicat.

Les cases suivantes (boutons radio) proposent trois choix exclusifs pour trianguler une base de données. Les trois méthodes seront exposées plus loin. Par défaut, c’est le choix 2 qui est utilisé.

Les cases suivantes (boutons radio) proposent les deux méthodes de rotation de la base de données à la souris. Le mode Direct agit seulement sur les axes X et Z de la base de données. Pour obtenir une rotation autour de Y, il faut passer par le second curseur en bas de l’interface. C’est le premier moyen de rotation qui a été introduit initialement dans OVNI, car c’est aussi le plus simple à gérer dans OpenGL. Le mode Trackball est plus récent. Il permet d’agir sur la projection des axes X et Y de l’écran dans le système d’axe de la base de donnée. On agit donc de façon indirecte sur les 3 axes à la fois, sauf dans les quelques cas où les axes écran et de la base de donnée coïncident. Néanmoins, dès qu’une légère rotation a été effectuée, les axes ne coïncident plus. Cette méthode de rotation est un peu plus difficile à maîtriser à l’usage que la méthode directe, mais elle est bien plus puissante, car autorise de positionner la base sous n’importe quels angles de présentation rien qu’à la souris, ce qui n’est pas possible par la méthode directe, sauf à passer par les 3 curseurs.

Les cases suivantes (boutons radio) proposent un choix de tailles d’icônes dans la barre d’outils. Par défaut, les icônes sont en 16x16 pixels. Sur les écrans « haute résolution », cette taille peut être un peu trop faible. D’autres choix sont proposés (pour le moment 24x24, 32x32 et 48x48, mais pourrait être étendu si besoin).

La case à cocher suivante permet d’afficher le nombre de trames par secondes obtenu lors de l’affichage de l’objet. En fait, cette option est intéressante dans des applications OpenGL ou Glut pures. Dans OVNI, wxWidgets ne pilote l’affichage que pendant les mouvements. Le nombre de trames par secondes n’a donc d’intérêt que lorsque l’objet est en mouvement, c'est-à-dire lors des déplacements et rotations, à la souris ou au clavier. Lorsque l’objet ne bouge pas, c’est la dernière valeur calculée qui est affichée.

La case à cocher suivante permet de conserver dans un fichier de sauvegarde (ou backup), en ajoutant .bak au nom courant, la base telle qu’elle était avant l’appel au menu « Enregistrer ». Cette option ne fonctionne que sur des bases de données de type Oktal (extension .bdd). Sur les autres types de bases (.3ds, .g3d, …), on passe forcément par le menu « Enregistrer sous … » et donc une sauvegarde n’est pas utile puisque le fichier original n’est pas touché. Par défaut cette case est cochée. En cas de problème lors de l’enregistrement de la base de données modifiée, on peut ainsi récupérer la version précédente de la base de données via ce fichier \*.bdd.bak, tout simplement en supprimant l’extension .bak dans le nom.

La case à cocher suivante n’est activée que si la création du fichier .bak est cochée. En la cochant, on force la suppression du fichier \*.bdd.bak quand on quitte Ovni. Par défaut, cette case n’est pas cochée. Si tout va bien, cela évite de polluer le répertoire de sortie avec des fichiers de sauvegarde. Toutefois, s’il y a eu un problème lors de l’enregistrement, problème dont on ne s’aperçoit pas forcément tout de suite, on perd le bénéfice de la sauvegarde. Option à ne cocher qu’avec précautions donc !

La boîte d’entrée du bas permet de spécifier le répertoire des fichiers temporaires de travail. Cela est utile pour les sauvegardes automatiques au cas où les répertoires par défaut ne seraient pas accessibles l’un et l’autre en écriture.

L’avant dernière zone du bas de la boîte de dialogue « Préférences » permet de paramétrer le nombre de threads ou tâches à effectuer en parallèles dans OpenMP. Typiquement, avec un processeur quadcore, les temps d’exécution de certaines boucles for peuvent être divisées d’un facteur 4 environ. Par défaut, la valeur de 0 indique à OpenMP d’utiliser le nombre maximum de threads. On peut choisir 1, 2, … jusqu’au nombre maximum de cœurs du processeur.

Le Bouton Reset, permet de donner les valeurs par défaut à l’ensemble des choix de cette fenêtre. Le bouton OK valide ces choix. À noter que la sauvegarde dans le fichier Ovni.ini ne sera effectivement réalisée qu’en sortie de programme.

Le sous-menu Hardware 3D permet d’afficher certaines des caractéristiques du hardware de la carte graphique utilisée. (Voir exemple ci-dessous).



Le denier sous-menu de Fichier est la fonction classique « Quitter ». On obtient la même chose en cliquant sur le X de fermeture en haut à droite sous Windows ou la combinaison de touches Alt-F4. Quand on quitte, OVNI sauvegarde automatiquement le fichier Ovni.ini si celui-ci a été modifié ou n’existait pas auparavant. De plus, en cas de modification de la base de données, OVNI propose de lancer la procédure d’enregistrement, qu’on peut accepter ou refuser.

#### Le menu Affichage

|  |
| --- |
|  |

Les 3 premiers sous menus peuvent être cochés de façon indépendante. Par défaut, « Plein » est coché. C’est ce qui permet de visualiser les surfaces des objets.

Le sous-menu « Points » permet d’afficher les points de la base de données sous forme de petits carrés bleus. Seuls les points utilisés par au moins une facette sont ainsi affichés. Les points non utilisés n’apparaissent pas.

Le sous-menu « Filaire » trace les arêtes des facettes. À noter que l’aspect des lignes peut être amélioré en utilisant l’Antialiasing géré par OpenGL (menu Fichier/Préférences).

La barre de boutons offre directement sous forme d’icônes ces trois choix, plein (plein), filaire (filaire) et points (points).

Le sous-menu « Axes » trace les 3 axes X, Y et Z de la base de données (icône axes).

Le sous-menu « Boîte englobante » trace les coins d’une boîte qui englobe les objets de la bdd. Ces coins sont déterminés à partir des minima et maxima sur chacun des 3 axes (icône boite).

Le sous-menu « Source de lumière » permet de tracer un petit cercle plein en jaune matérialisant la position d’une source d’éclairage pour OpenGL. Dans OVNI, il n’y a qu’une seule source de lumière (icône sun).

Les 4 sous-menus suivants, « Origine », « Vue de face », « Vue de profil » et « Vue de dessus » sont quatre postions prédéfinies d’observation. À noter que les notions de dessus, dessous, sont définies pour un système d’axes standard dans Crira. Si une base de données est construite différemment, les appellations peuvent ne pas correspondre.

Les 4 derniers sous-menus permettent d’ouvrir des boîtes de dialogues dans lesquelles on pourra entrer des valeurs spécifiques.

|  |  |
| --- | --- |
|  |  |
|  |  |

Les fenêtres comportent éventuellement un bouton Appliquer. C’est en effet nécessaire si on entre directement une valeur numérique dans une des sous-fenêtres. Il faut valider cette valeur pour qu’elle soit prise en compte par wxWidgets. Si on utilise les petites flèches en bord de sous-fenêtre, la validation est automatique.

Les angles sont donnés en degrés. Par convention, LSI est l’angle de site avec le plan horizontal XY. LAZ est l’angle d’azimut. Il vaut 0° pour une vue plein avant, 180° pour une vue plein arrière, si LSI est à 0°. Dans le système d’axes standard, LAZ=0 et LSI=0 correspond à des rotations autour de X de 270° et nulles autour de Y et Z. Dans une vue de dessus les rotations en X et Y sont nulles et celle en Z est de 270°. Si les fenêtres sont ouvertes, toute modification des angles avions faite par ailleurs est mise à jour automatiquement dans les fenêtres, que ce soit à la souris, au clavier, avec les curseurs. Attention, les valeurs d’angles en site et azimut ne sont valables que si la rotation en Y est nulle (⬄ angle de roulis de l’avion = 0 ⬄ vol à plat, hors virage). Si ce n’est pas le cas, un message de warning est affiché.

#### Le menu Primitive

|  |
| --- |
|  |

Les divers sous-menus permettent d’ajouter des objets génériques simples à la base de données déjà affichée. Chaque sous-menu affiche une boîte de dialogue spécifique permettant d’entrer divers paramètres de la forme générique.

Si une ou plusieurs formes génériques ont été ajoutées, le sous-menu du bas permet de les supprimer une à une dans l’ordre inverse des créations.

Pour une sphère ou un ellipsoïde, il faudra entrer un nombre de méridiens et de parallèles. Plus ce nombre est élevé, plus la forme sera précise. A l’inverse, s’il y en a très peu, comme par exemple 4 méridiens et 1 seule parallèle, on obtiendra un cube avec des axes décalés de 45°. Deux types de tracés sont possibles. Le premier est celui qui existait en version Tcl. Le second, sélectionnable via une case à cocher, donne un rendu plus régulier, plus symétrique, sans direction privilégiée. Bien qu’il fonctionne avec n’importe quel nombre de parallèles et méridiens, la symétrie de construction ne sera bonne que pour un nombre pair de méridiens.

|  |  |
| --- | --- |
| Version originale | Nouvelle version |

Pour obtenir un parallélépipède, il faut dans cette version, d’abord créer un cube qu’on déformera ensuite.

L’interface type de génération d’une primitive se présente ainsi :



On y propose, ici dans le cas d’un cylindre, des valeurs par défaut pour divers paramètres géométriques ainsi que les entrées en numéro de groupe (et matériau) pour Crira.

L’icosaèdre est une forme géométrique régulière composée de 12 triangles équilatéraux égaux. En soi, cette forme n’a pas beaucoup d’intérêt, mais lorsqu’on subdivise chaque triangle en 4 sous-triangles, qu’on ramène les sommets créés sur une sphère, le tout de façon récursive, on obtient une approximation de sphère un peu meilleure que celle de la primitive spécifique. A noter toutefois que la notion de méridiens et parallèles n’y existe pas, ce qui ne convient pas forcément à toutes les études.

|  |  |
| --- | --- |
| Icosaèdre de base | Icosaèdre avec 1 subdivision |
| Icosaèdre avec 2 subdivisions | Icosaèdre avec 4 subdivisions |

Le nombre de subdivisions est limité à 8 car, au-delà, on a des soucis de capacité mémoire, même en version 64 bits. Le nombre total de facettes obtenu avec 8 subdivisions est de 1 310 720 ! Le nombre total de sommets créés par la procédure récursive est de 1 310 712, sans tenir compte du fait qu’un sommet est utilisé plusieurs fois (*le nombre de sommets nécessaires, obtenus après simplification de la base de données est en fait de 655 362*). Toutefois, déjà avec 2 subdivisions, surtout avec lissage de Gouraud, l’approximation d’une sphère est très bonne. *Note* : il est envisagé de créer directement le nombre de sommets nécessaires dans une version future. Toutefois, la procédure envisagée risque d’être longue à l’exécution, du fait de la récursivité de la génération. Cela demande donc un peu de réflexion.

Un tore peut être utilisé pour simuler une chambre à air, un pneu. Les données principales sont le rayon moyen et le rayon du cercle générateur. Le rayon global externe du tore est la somme de ces deux valeurs et le rayon global interne est leur différence. Chaque secteur est composé d’un cercle générateur décomposé en plusieurs points.

#### Le menu Repérage

##### Repérage : Premier groupe de sous-menus

|  |
| --- |
|  |

Les 5 premiers sous-menus affichent des boîtes de dialogues permettant de visualiser un point, une facette, un groupe, un matériau ou un objet en entrant son numéro dans la fenêtre appropriée.

Pour un point ou une facette, il faut également entrer le numéro d’objet auquel il appartient.

Un objet ainsi sélectionné apparaît en rouge dans la fenêtre OpenGL de visualisation.

Les matériaux ou les groupes seront identifiés en bleu en choisissant leur numéro à l’aide des flèches de droite. Si des numéros sont absents de la base de données, ils sont automatiquement sautés.

Les fenêtres « Point » et « Facette » permettent en plus d’effectuer un certain nombre de manipulations.

|  |
| --- |
|  |

La fenêtre « Repérage d’un point » affiche les coordonnées en x, y et z du point sélectionné. Dans la fenêtre OpenGL, le point est matérialisé par une étoile violette.

On peut modifier les coordonnées du point en x, y ou z en cochant la (ou les) case(s) de droite correspondant à la (ou aux) coordonnée(s) à changer. Une fois entrée la valeur, on valide en cliquant sur le bouton « Modifier ». Les valeurs qui sont en grisé ne sont pas modifiables ou inactives. Elles le deviennent si la case à cocher correspondante est validée.

Le bouton « Undo » permet de revenir à la valeur qu’il y avait avant la modification.

Il est possible de centrer la rotation de la base de donnée sur le point sélectionné en cliquant sur le bouton prévu à cet effet. Utile pour regarder de plus près une partie de la bdd en y effectuant des zooms, des déplacements autour du point sélectionné.

En standard, quand on ferme cette fenêtre par le bouton « Quitter » ou la croix en haut à droite sous Windows ou Linux, l’étoile matérialisant le point s’efface. On peut la conserver en cochant la case adéquate prévue à cet effet.

Le bouton valider, qui était présent en version Tcl/Tk, n’existe plus dans cette version.

|  |
| --- |
|  |

La fenêtre « Repérage d’une facette » affiche ou permet de modifier certaines des caractéristiques d’une facette. Comme précédemment, la facette est donnée par son numéro de facette et le numéro d’objet auquel elle appartient. La facette est identifiée en jaune.

Pour chaque facette sélectionnée, on affiche son nombre de sommets. Si la case « Visualisation des sommets de la facette » est cochée, les sommets sont matérialisés par de petits carrés verts.

On peut alors explorer chacun des sommets à l’aide du bouton fléché donnant le numéro du point dans la facette. Le numéro du point dans la base de donnée et pour l’objet en question s’affiche alors dans la parte droite. Le point sélectionné est matérialisé par une étoile violette.

Les deux cases à cocher suivantes permettent de visualiser la normale à la facette par un petit trait en vert et/ou les normales aux sommets par de petits traits bleus. Ces deux choix sont conservés à la fermeture de la fenêtre. Ainsi, si on sélectionne une facette par un autre moyen, le/les normales seront aussi affichées. On peut modifier la longueur de ces normales dans le menu Fichier/Préférences.

À noter que ces cases à cocher ont leur équivalent sur la barre de bouton, respectivement avec les icônes  et .

Les boutons du dessous permettent :

* D’inverser la normale de la facette ;
* De supprimer la facette (il y a un bouton Undo). En réalité, on marque la facette comme « à supprimer ». Elle ne le sera réellement qu’en sortie du programme ou à l’enregistrement ;
* De masquer la facette. Une facette masquée n’est pas supprimée. Elle est toujours là et ne sera pas enlevée à l’enregistrement. Le bouton change d’intitulé en cas de masquage et propose de réafficher toute facette précédemment masquée ;
* D’effectuer une permutation circulaire des sommets. C’est une option utile si une facette à plus de trois sommets et que ceux-ci sont mal configurés pour un calcul de normale à la facette, comme par exemple 3 points alignés ;
* De centrer la rotation de la bdd sur le barycentre de la facette. Utile pour regarder de plus près une partie de la bdd en y effectuant des zooms, des déplacements autour de la facette sélectionnée.
* Le bouton « Reset » remet l’interface à ses valeurs originales.

Comme précédemment, quand on ferme la fenêtre la facette sélectionnée n’est plus affichée, sauf si on coche la case prévue à cet effet.

##### Repérage : Second groupe de sous-menus

Le second groupe de sous-menu de la fenêtre « Repérage » n’a qu’un seul élément : « Sens des normales ». C’est une fonction permettant d’identifier les facettes mal orientées en rouge. Pour OpenGL, vues depuis l’utilisateur, il existe 2 types de facettes : celles de type GL\_CW et celles de type GL\_CCW, correspondant respectivement au sens de parcours des sommets dans le sens des aiguilles d’une montre (Clockwise ou GL\_CW) et dans le sens inverse (Counterclockwise ou GL\_CCW). Une facette est correctement orientée si elle est de type GL\_CCW (c’est en fait le sens trigonométrique direct). Quand on active cette fonction, on dessine en fait deux fois l’objet : une fois en rouge en n’affichant que les facettes GL\_CW et une seconde fois en bleu en n’affichant que les facettes GL\_CCW. On repère ainsi facilement les facettes mal orientées, du moins celles dont le sens de parcours des sommets est inverse de ce qu’il devrait être pour une élimination optimale des facettes cachées, car la normale au barycentre intervient également dans la visualisation.

Si cette fonction est active, le menu correspondant apparaît comme coché.

On peut également activer cette fonction par un raccourci clavier via la lettre n.

##### Repérage : Troisième groupe de sous-menus

Le troisième groupe de sous-menu de la fenêtre « Repérage » a trois éléments :

* Couleurs facettes : colorise chaque facette avec une couleur différente. Existait dans la version Tcl/Tk mais comme elle ne présente que peu d’intérêt elle est désactivée ici (conservée pour le cas où on y retrouverait un intérêt, mais le code correspondant n’a pas été transposé) ;
* Couleurs groupes : colorise chaque facette avec une couleur dépendant du numéro de groupe au sens Crira. On obtient la même par le bouton  de la barre de boutons (En version Tcl/Tk on avait aussi la touche g (ou G), non transposée ici) ;
* Couleurs matériaux : colorise chaque facette avec une couleur dépendant du numéro de matériau. On obtient la même fonction par le bouton  de la barre de boutons (En version Tcl/Tk on avait aussi la touche M, non transposée ici).

Ces fonctions sont exclusives entre-elles. Dans les menus, la fonction active apparaît comme cochée.

#### Le menu Image

|  |
| --- |
|  |

Il comporte trois options d’enregistrement d’images.

La fenêtre OpenGL est alors enregistrée dans son état visuel courant dans le répertoire de travail parmi les 3 formats proposés, jpeg (.jpg), png (.png) ou ppm (.ppm). Le nom est imposé Image\_Ovni. Le répertoire est par défaut celui où se trouve l’exécutable d’OVNI mais ce peut-être aussi un autre répertoire de travail entré via le menu Fichier/Préférences.

Par défaut, à l’ouverture d’OVNI, la taille de l’image est de 796\*396 pixels sous Windows. Elle peut être plus grande si on a redimensionné la fenêtre globale, soit à la souris, soit via le mode plein écran. De même, si les curseurs du bas sont masqués.

#### Le menu Déplacement

Ce menu a disparu dans cette version car, à l’usage, il s’avère sans intérêt vu qu’on obtient la même chose plus simplement à la souris ou au clavier.

#### Le menu Outils

|  |
| --- |
|  |

C’est dans cette section qu’on trouve les boîtes de dialogues permettant de modifier certains éléments de la base de données. Les 3 premiers sous-menus n’existaient pas en version Tcl/Tk. Ils permettent d’effectuer quelques opérations de bases sur des facettes sélectionnées directement depuis la fenêtre OpenGL. Ils ont leur équivalent dans le menu contextuel obtenu via un clic droit dans la fenêtre OpenGL.

Les deux sous-menus suivant sont les plus complexes. Ils font chacun l’objet d’un sous-paragraphe détaillé ci-dessous. Ces deux sous-menus étant fréquemment utilisés, un accès direct par icônes est prévu dans la barre d’outil, respectivement outils et modifs.

Le sous-menu « Sélections et déplacements », permet de modifier certains éléments spécifiques comme des points, des facettes, des objets. Le sous-menu « Modifications », va permettre de créer des points, d’effectuer des « soudures » et aussi d’effectuer des modifications plus globales sur la base de données. Ces sous-menus conduisent à des actions interactives qui seront pilotées à la souris.

Le sous-menu « Choix des objets à afficher » propose sous la forme d’une liste les divers objets de la base de données. Seuls sont affichés ceux dont la case à cocher en regard est activée. Pour aller plus rapidement, les boutons du bas de la fenêtre, « Tous » et « Aucun », sont prévus à cet effet.

Le sous-menu « Réafficher les éléments masqués » permet de réafficher en un seul clic, tous les éléments, facettes et objets, qui ont été masqués un à un via les divers menus.

Le sous-menu « Supprimer les éléments masqués » permet de supprimer de la base de données tous les éléments, facettes et objets, qui ont été masqués. En réalité, ces éléments sont marqués comme à supprimer et ne le seront effectivement qu’à l’enregistrement de la base de données.

Le sous-menu « Restituer les éléments supprimés » permet justement, tant que l’enregistrement n’a pas été effectué, de restituer tous les éléments supprimés (objets, facettes).

##### Sous menu Sélections et Déplacements

|  |
| --- |
|  |

Le mode de sélection permet de choisir une des 3 modes, Points, Facettes ou Objets. Par défaut, à l’ouverture, c’est le mode « Facettes » qui est sélectionné. A noter, que c’est aussi ce mode « Facettes », qui est sélectionné par défaut dans la fenêtre OpenGL et dont certaines fonctions sont directement accessibles via le menu contextuel obtenu à partir d’un clic droit à la souris.

Un certain nombre de bouton ne sont pas actifs suivant le mode de sélection ou même changent d’intitulés afin d’offrir à l’utilisateur une possibilité en relation avec les actions précédentes.

La sélection est surtout prévue pour être pilotée à la souris, via des clics sur le bouton du milieu (ou la molette) sur les points, facettes ou objets à sélectionner. On a toutefois la possibilité d’effectuer une sélection manuelle avec le premier bouton.

En mode sélection de facette, on aura alors la possibilité d’entrer directement les numéros de facettes à sélectionner en présélectionnant l’objet à l’aide d’un bouton à flèches. Les numéros entrés doivent être séparés par des espaces ou des virgules.

La sélection manuelle est désactivée en mode Point.

En mode Objet, c’est une fenêtre avec des cases à cocher qui est proposée. Les ou les objets cochés seront sélectionnés. Pour aller plus rapidement, deux boutons, intitulés « Aucun » et « Tous », sont prévus à cet effet.

Si on veut repasser en mode de sélection par clic de souris après avoir utilisé le mode manuel, il faut recliquer sur le mode de sélection désiré.

Le second bouton, dont l’intitulé s’adapte en fonction du choix de sélection, permet d’accéder directement aux fenêtres des menus Repérage/Points, Repérage/Facettes et Repérage/Objets. Si ces fenêtres secondaires sont ouvertes, elles sont automatiquement mises à jour à chaque clic de souris sur un élément sélectionné (fonction du mode de sélection choisi).

La sélection à la souris s’effectue par un clic sur le bouton milieu. Si on veut sélectionner plusieurs éléments (sélection multiple), il suffit de cliquer successivement sur les éléments à sélectionner. Ces clics fonctionnent comme des bascules : un élément non sélectionné devient sélectionné et réciproquement (*Note : en version Tcl/Tk, il fallait en plus utiliser la touche Ctrl : option non reconduite ici*).

En mode Sélection de points, les points sont affichés par de petits carrés bleus. La couleur d’un point survolé passe en jaune dès qu’on est suffisamment près. Une fois cliqué, sa couleur passe en vert. (*Note : dans le code en version wxWidgets, il a été nécessaire d’introduire un petit offset de 4 pixels sur la position en Y du pointeur de la souris, sinon le sommet de la flèche du pointeur était décalé par rapport à la sélection effective. Très sensible en sélection de points ou d’arêtes*).

Le groupe « Informations » affiche des informations sur le ou les éléments sélectionnés. En version Tcl/Tk et en mode facette (ou en mode Point), seules étaient sélectionnables, en cas de sélection multiple, les facettes (ou les points) appartenant au même objet que la facette (ou le point) sélectionnée en premier. Cette restriction n’a pas été reconduite ici. Le numéro de facette, de groupe ou de matériau correspond à la dernière facette cliquée.

L’ensemble des cases et des boutons sous l’intitulé « Modifier la sélection » permet, comme son nom l’indique, de modifier certains éléments sélectionnés.

Les deux boutons radio, Groupe et Matériau, sont exclusifs. Si le premier est coché, on visualise et éventuellement modifie des numéros de groupe. Si c’est le second qui est coché, ce sera sur des numéros de matériau qu’on fera la même intervention. La fenêtre située sous les boutons radio affiche les numéros de groupes (ou de matériaux) effectivement utilisés dans la bdd. Toute modification y est répercutée au fur et à mesure. Ainsi, si on supprime la dernière facette qui a un numéro de groupe 4 par exemple, la valeur 4 est aussitôt supprimée de la liste affichée.

En mode Sélection de facette ou Sélection d’objet, on peut entrer un numéro de groupe (ou de matériau) dans la case prévue à cet effet. Toutes les facettes (ou objets) sélectionnées possèderont ce nouveau numéro de groupe (ou de matériau) après qu’on ait validé l’action par le bouton « Appliquer » situé à côté de cette case.

La case « Forcer le mode facettes planes » n’est, en principe, disponible qu’en mode Sélection d’objet. Elle doit permettre, pour l’objet sélectionné, de forcer à ce que les facettes de cet objet soient considérées comme planes, et donc de court-circuiter le lissage de Gouraud obtenu à l’aide des normales aux sommets. Elle était active en version Tcl/Tk, mais ne l’est pas encore en version wxWidgets (code non transposé).

L’intitulé des boutons qui suivent est suffisamment explicite. En fonction du choix de sélection et/ou des actions passées, l’intitulé peut changer ou le bouton être inactif (intitulé grisé), l’action n’ayant alors pas de sens. On peut donc :

* Inverser les normales des facettes ou des objets sélectionnés. Cette action inverse également le sens de parcours des sommets de facettes ;
* Supprimer la/les facettes sélectionnées ou Supprimer l’objet sélectionné (la sélection peut ici aussi être multiple). Ces 2 premiers boutons possèdent une fonction Undo accessible par le bouton de droite ;
* Fusionner les objets sélectionnés. Ce bouton ne s’active qu’en mode de sélection d’objets et si au moins 2 objets sont sélectionnés. Le premier objet sélectionné est l’objet de base. Les points, facettes, normales … des autres objets y seront ajoutés en augmentant les tailles des tableaux (ou vecteurs) internes de l’objet de base. Les objets autres que le premier sélectionné sont supprimés. Il peut être utile, mais non nécessaire, de faire suivre cette action pas une simplification de la base de donnée, ce qui éliminera les points, normales en doublon (voir le menu ou le bouton « Modifications »).
* Inversion du sens de parcours des facettes. Cette action est valide en mode Sélection de facettes mais aussi en mode Sélection d’objets. Dans ce dernier cas, toutes les facettes des objets sélectionnés seront impactées. Cette action se caractérise par inverser le sens dans lequel sont parcourus les sommets de la facette. Les normales ne sont pas touchées ;
* La permutation circulaire des sommets existait en version Tcl/Tk mais n’a pas été reconduite ici car jugée inutile à ce niveau. Elle existe en sélection individuelle via clic droit de la souris.
* Masquer les facettes ou Masquer l’objet. Opère sur la sélection multiple. Si une facette ou un objet vient d’être masqué, l’intitulé du bouton change et on peut le réafficher aussitôt ;
* Tout réafficher, permet comme son nom l’indique de réaffiche tous les éléments qui ont été masqués, et pas seulement le dernier ;
* Centrer la rotation sur l’Objet/les Objets, les facettes ou les points : permet de centrer la rotation de la base de donnée sur le barycentre des objets, facettes ou points sélectionnés ;
* Le bouton Manipulation de l’objet est accessible dans tous les cas de Sélection. Il ouvre une sous-fenêtre permettant d’effectuer des translations, des rotations, des mises à l’échelle d’éléments, ou encore de créer de nouveaux objets par symétrisation d’objets existants. Seules les modifications, valides pour le mode sélection choisi, sont proposées (le bouton des fonctions non valides est inactif et son intitulé reste grisé) ;
  + En mode Sélection d’objet, on peut translater le (ou les) objets, effectuer une rotation sur un (ou des) objets(s), faire une mise à l’échelle. Chaque action ouvre une sous-fenêtre supplémentaire. Certaines entrées demandent à être validées par un clic sur le bouton adéquat avant d’avoir un effet sur l’objet affiché, d’autres pas (c’est le cas quand on utilise les boutons fléchés, l’action étant alors immédiate). On peut aussi créer un objet par symétrisation sur un axe, 2 axes ou les 3. L’effet est d’abord symbolisé par une petite boîte englobante colorée montrant l’objet d’origine et celui qui sera créé. Ce symbolisme est créé en temps réel, dès qu’on coche une des cases X, Y ou Z. Une fois que l’action désirée est correctement identifiée, on la valide en cliquant sur le bouton « Créer l’objet ». Ce n’est qu’à ce moment que l’objet est réellement créé par la symétrie choisie ;
  + En mode Sélection de facette, la rotation est désactivée. Les autres fonctionnent comme pour un objet ;
  + En mode Sélection de Point, c’est la même chose. À noter que la mise à l’échelle n’a de sens que si plusieurs points sont sélectionnés. Dans ce cas, la réduction ou l’augmentation d’échelle se fait autour du barycentre des points sélectionnés. Par exemple, sur un objet de type cylindre, on peut sélectionner l’ensemble des points sur l’un des cercles extrémité, puis par mise à l’échelle faire évoluer le cylindre vers une forme conique, voir même un cône avec un sommet, si le facteur d’échelle est nul.
  + Les différentes sous-fenêtres proposées sont les mêmes quel que soit le mode de sélection :

|  |
| --- |
|  |

La fenêtre « Translation » propose deux pas de progression, un pas rapide (ou grossier) à gauche et un pas fin à droite. En cas de modification d’un des pas, il faut d’abord soit le valider via la touche adéquate, soit tout simplement utiliser la touche « Entrée » du clavier une fois entrée la nouvelle valeur. Ensuite, on peut effectuer les translations sur chacun des axes séparément, l’action sur les boutons fléchés modifiant directement l’élément sélectionné (point, facette, objet, sélection simple ou multiple) en utilisant comme incrément le pas correspondant. L’amplitude de variation des valeurs fines de droite s’ajuste automatiquement au pas rapide : Valeur rapide ± Pas rapide. Dans le cas ci-contre, une translation fine en X peut être faite entre -0.2 et +0.2 par pas de 0.01. Ici aussi, une entrée directe, validée par la touche « Entrée » au clavier est possible dans les fenêtres « Translation » de gauche (mais pas dans celles de droite, grisées).

|  |
| --- |
|  |

La fenêtre « Rotation » n’est disponible que pour une sélection en mode Objet. Le pas est entré via la fenêtre du haut. Le centre de rotation par défaut est le barycentre de l’objet. On peut choisir également le centre de la boîte englobante de la sélection (pour des objets bien échantillonnés dans les 3 axes, la différence est minime avec le choix par défaut) ou encore entrer explicitement les coordonnées x, y et z du centre de rotation. Penser à valider avant d’utiliser ce centre. L’objet sera d’abord manipulé graphiquement en temps réel dès qu’on clique sur les boutons fléchés en utilisant les fonctions de rotations intégrées à OpenGL sur chacun des 3 axes. En cliquant sur le bouton « Appliquer » du bas, on modifie explicitement les coordonnées des points dans la base de données sur le ou les objets sélectionnés. En raison des spécificités d’OpenGL, et notamment de l’ordre dans lequel on effectue ces rotations, on peut ne pas obtenir le même résultat si on effectue toutes les rotations à la fois ou seulement axe par axe. Il vaut mieux procéder axe par axe et valider chaque axe indépendamment pour obtenir au final l’effet voulu.

|  |
| --- |
|  |

La fonction mise à l’échelle fonctionne sur le même principe que les 2 autres. Le pas d’incrémentation de l’échelle, s’il est modifié, doit d’abord être validé, soit via le bouton « Valider » soit via la touche « Entrée » du clavier. La mise à l’échelle peut être commune sur les 3 axes (case à cocher du bas) ou indépendante sur les 3 axes. C’est ainsi qu’on peut transformer un cube en parallélépipède rectangle. Cette fonction utilise des ressources graphiques d’OpenGL. L’action sur la base de données n’est effective qu’après avoir cliqué sur le bouton « Appliquer » du bas.

Le dernier groupe de fonction agit sur le type de sélection des facettes ou des points. Ici aussi, seules sont proposées les actions valides pour le mode de sélection retenu. ***Ces options ne sont que partiellement disponibles pour le moment, en version wxWidgets. Il est prévu de les réintroduire, c’est pourquoi le texte ci-dessous les mentionne.***

~~En mode sélection de point, deux possibilités sont offertes : le mode « Les 2 faces » ou le mode intitulé « Face avant/Face arrière ». En mode « Les 2 faces », on sélectionne les points un à un (clic sur un point avec le bouton milieu de la souris). On obtient une sélection multiple en cliquant sur un nouveau point (~~*~~un clic sur un point déjà sélectionné le désélectionne~~*~~). Un point survolé passe en jaune dès qu’il est cliquable. C’est le mode proposé par défaut. En mode dit « Face avant », on dessine un petit rectangle grisé en cliquant sur le bouton milieu de la souris puis en glissant la sélection (déplacement de la souris en gardant le bouton du milieu enfoncé). Tous les points à l’intérieur du rectangle sont sélectionnés. Attention, dans ce mode, les points cachés par l’objet sont aussi sélectionnés.~~

En mode sélection de facette, trois possibilités sont offertes :

* le mode « Les 2 faces » qui permet de sélectionner les facettes à la souris par clic sur le bouton du milieu. Une sélection multiple est obtenue en cliquant plusieurs facettes ou en déplaçant le curseur tout en maintenant le bouton enfoncé . C’est le mode par défaut ;
* le mode « Faces avant » qui permet de sélectionner un ensemble de facettes dont la normale est orientée vers l’utilisateur. ~~Un clic sur le bouton du milieu de la souris puis un glissé de la souris dessine un rectangle grisé.~~ ~~Toutes les facettes dont au moins un sommet est à l’intérieur du rectangle et dont la normale est correctement orientée seront sélectionnées.~~ Attention si un objet en cache un autre. Les facettes correctement orientées dans l’objet caché seront aussi sélectionnées ;
* Le mode « Faces arrière » est équivalent au précédent, mais ce sont les facettes dont la normale n’est pas orientée vers l’utilisateur qui seront sélectionnées. Même précautions que pour l’autre mode : attention aux facettes cachées.

##### Sous menu Modifications

|  |
| --- |
|  |

Ce sous-menu est utilisé, soit pour créer les points sur une arête de facette, soit pour créer des facettes individuelles, soit pour « souder » deux points. La partie « Divers » en bas offre des fonctions générales sur l’ensemble de la base de données.

**Ajout de points dans une arête :**

Quand on clique sur le bouton « ***Ajouter sous le pointeur*** » on obtient un mode de sélection des arêtes. On reste dans ce mode tant que le bouton reste enfoncé. Le mode d’affichage filaire est automatiquement activé. Dans ce cas, dès qu’on survole à la souris une arête, c'est-à-dire un bord de facette, celle-ci s’affiche en vert. Si on clique avec le bouton milieu de la souris sur cette arête, un point est créé à l’emplacement du clic. La facette ainsi modifiée comportera un point de plus. On sort de ce mode en cliquant de nouveau sur le bouton « ***Ajouter sous le pointeur*** ».

Le second bouton, « ***Diviser en*** » est une variante de ce type de sélection. Dans ce cas, l’arête sur laquelle on clique sera divisée en autant de segments de longueur égale que ce qui est indiqué à droite du bouton. La valeur du nombre de segments peut être modifiée par les flèches à droite de la fenêtre d’entrée. Les points intermédiaires sont créés, augmentant ici aussi le nombre de points dans la facette. Par construction, les points créés sont alignés. Cela peut poser des problèmes dans les calculs de normales. En principe, OVNI détecte cela et ne tient pas compte de ces points, mais …

L’utilisation de cette fonction est utile lorsque on a une base de données dont certaines facettes sont en principe jointives, mais dans la réalité, et pour peu qu’on utilise un zoom adéquat, présentent des espaces inter-facettes.

En particulier, au centre de l’image ci-dessous, on peut constater que l’extrémité de triangles de la partie gauche arrive sur le milieu d’une arête d’une facette de droite. Le point commun des sommets des triangles peut ne pas être exactement sur l’arête de la facette de droite.



La facette de droite étant plane, si le sommet des triangles n’est pas exactement sur l’arête, on peut avoir un petit interstice entre la facette de droite et les facettes de gauche. Cet interstice sera visible dans certains cas d’observation, plus ou moins tangentiels aux facettes

Dans ce cas, une solution consiste à créer dans la facette de droite un point supplémentaire sur l’arête. Pour s’assurer que le point ainsi créé ne produit pas de trou, on en fera une « soudure » avec le sommet commun des triangles de gauche (voir un peu plus loin cette option).

**Création de facettes :**

Quand on clique sur le bouton « ***Créer*** » placé dans cette rubrique, on passe dans un mode de création de facettes à partir de sommets existants. Tant qu’on est dans ce mode, le bouton « Créer » reste enfoncé. Le mode d’affichage des points est automatiquement activé. La facette sera créée avec les numéros de groupe et de matériau entré dans les fenêtres de cette rubrique. Il suffit ensuite de cliquer dans l’ordre voulu sur les différents sommets qui doivent composer la nouvelle facette. Une fois le premier point sélectionné (clic avec le bouton milieu de la souris), il passe en vert. Au survol, les points suivant deviennent jaunes et passent en vert si on les sélectionne par clic. Dans ce mode, les points peuvent appartenir à un même objet ou à des objets différents. C’est le premier point cliqué qui fixera l’appartenance de la facette à un objet. Si tous les points sont dans le même objet, tous les points utiles existent déjà. S’ils sont dans un autre objet, des points supplémentaires, copies des points originaux des autres objets, seront ajoutés au premier objet. La facette sera créée, avec les numéros de groupe et de matériau entré, dès lors qu’on cliquera de nouveau sur le premier point sélectionné pour clore cette phase de création. La facette créée apparaît alors en vert clair si la normale est orientée vers l’observateur, ou en vert foncé dans le cas contraire. ***Remarque*** : pour avoir une facette orientée vers l’utilisateur, tourner dans le sens trigonométrique, sinon, tourner dans le sens des aiguilles d’une montre. En cas d’erreur, on peut annuler la création en cours en cliquant sur le bouton prévu à cet effet.

En cas d’erreur, le bouton du dessous permet de supprimer la facette qui vient d’être créée. Enfin, si on s’est trompé de sens de rotation, le dernier bouton de la rubrique est là pour inverser le sens de la normale à la facette.

La facette ainsi créée appartiendra au même objet que tous les sommets sélectionnés.

On quitte ce mode en cliquant de nouveau sur le bouton « ***Créer*** ».

**Souder deux points :**

Cette fonction est utilisée pour forcer deux points à être confondus. On amène le premier point sélectionné à être confondu avec le second point sélectionné. Quand on passe dans ce mode, en cliquant sur le bouton « ***Souder*** » on passe en mode d’affichage des points et les points bleus passent en jaune quand on les survole. Le bouton « Souder » reste enfoncé. On clique alors sur le premier point pour le sélectionner, puis on l’amène sur le second point qui n’est pas forcément dans le même objet.

Si les deux points appartiennent au même objet, le premier point sera effectivement remplacé par le second dans toutes les facettes qui l’utilisent. C’est seulement un changement d’indice de point qui est opéré dans les facettes qui l’utilisent. Le premier point ne sera plus utilisé, bien qu’il soit toujours présent dans le tableau des points de la base de données.

Si les deux points appartiennent à des objets différents, on remplace les coordonnées x, y et z dans l’objet auquel appartient le premier point sélectionné, par les coordonnées correspondantes du second point sélectionné dans le deuxième objet. Ici, les indices de points des facettes ne changent pas. Les normales aux facettes qui utilisent le point modifié, (le point soudé donc) sont recalculées par OVNI.

On quitte ce mode en cliquant de nouveau sur le bouton « Souder ».

**Undo :**

On peut annuler les soudures ou les divisions d’arêtes en cliquant sur le bouton « ***Undo*** » situé en dessous. La fonction Undo est commune aux deux options et est cumulative. Elle efface les modifications une à une, dans l’ordre inverse où elles ont été effectuées.

**Divers :**

Les fonctions de cette rubrique s’appliquent à toute la base de données.

La fenêtre « ***Tolérance d’égalité*** » permet d’entrer une valeur flottante qui sera utilisée dans les tests d’égalité entre deux points. La valeur est entrée en pourcentage de la diagonale de la boîte englobante. Ces tests sont notamment utilisés dans la « Simplification de la BDD » ci-dessous. Cette tolérance sert, dans un objet donné, à déterminer si deux points différents, peuvent en fait être considérés comme confondus car leur distance est inférieure à la tolérance. On peut ainsi réduire le nombre de points utilisés dans une base de données.

Le bouton « ***Inverser toutes les normales*** » est l’équivalent de ce qu’on trouve dans le menu Transformation qu’on verra ci-dessous. Il inverse toutes les normales de toutes les facettes de tous les objets. Comme le sens d’une normale dépend du sens de parcours des sommets, il arrive qu’il faille cliquer 2 fois sur ce bouton pour en voir l’effet : la première fois, on corrige une non-conformité entre le sens de parcours des sommets et la normale au barycentre, le second clic donne le bon résultat.

Le bouton « ***Trianguler la BDD*** » permet de transformer en triangles les facettes de la bdd qui ont plus de 3 sommets. Il y a donc création de facettes, la facette originale étant supprimée. Trois méthodes de triangulation sont proposées via le menu Fichier/Préférences. La méthode 1 utilise le premier sommet de la facette comme sommet commun à tous les triangles générés. Son inconvénient est de donner plus d’importance à ce sommet qu’aux autres. Néanmoins, pour des facettes quasi planes, le résultat est satisfaisant. La méthode 2 utilise alternativement les numéros de sommets croissants et les numéros de sommets décroissants pour créer les différents triangles. Ainsi, sur une facette à n sommets, les triangles sont générés ainsi : (1,2,3), (1,3,n), (n,3,4), (n,4,n-1), (n-1,4,5), … Cette méthode donne également de bons résultats, mais comme la première, elle ne garantit pas que dans certains cas particuliers on ne tombe pas sur une génération de triangle à partir de 3 points alignés, ce qui pose problème pour les calculs de normales. La méthode 3 enfin, génère d’abord un point supplémentaire au barycentre de la facette puis se sert de ce nouveau point comme sommet commun à tous les triangles générés. Cette méthode ne donne pas de triangles dégénérés (3 points alignés), ne donne pas plus de poids à un sommet plutôt qu’à un autre et si la facette n’est pas plane, c’est celle qui approxime le mieux la surface. Son inconvénient est d’alourdir la base de donnée par augmentation du nombre de points et elle crée 2 facettes de plus que les méthodes précédentes pour chaque facette triangulée. Les figures ci-dessous démontrent à partir d’une facette originale à 10 sommets les 3 méthodes de triangulation. Les méthodes 1 et 2 créent 8 facettes, la méthode 3 en crée 10. La numérotation des sommets utilisée dans ce cas est indiquée avec la méthode 2 (chiffres en noir). La numérotation des triangles dans l’ordre de création est indiquée en rouge.

|  |  |
| --- | --- |
| image  Facette originale | image  **8**  **7**  **6**  **5**  **4**  **3**  **2**  **1**  Méthode 1 |
| image  **5**  **3**  **2**  **8**  **7**  **6**  **4**  **1**  **4**  **6**  **1**  **10**  **9**  **8**  **7**  **5**  **3**  **2**  Méthode 2 | image  **10**  **9**  **8**  **7**  **6**  **5**  **4**  **3**  **2**  **1**  Méthode 3 |

C’est la méthode 2 qui est programmée par défaut. Le choix est conservé dans le fichier Ovni.ini.

La case à cocher « **Forcer facettes à NON planes** » s’utilise lors du Recalcul des Normales ci-après. C’est une recopie de la case équivalente dans le menu Préférences lors de l’ouverture de la boîte de dialogue. Il agit sur tous les objets alors que la case à cocher de la boîte de dialogue « Sélection et déplacement » n’agit que sur la sélection en cours (objets ou facettes). Toutefois, la valeur utilisée ici ne sera que temporaire, et contrairement à ce qui se passe via le menu Préférence, sa modification ne sera pas enregistrée dans le fichier init. Pour rappel, une facette peut être déclarée plane et enregistrée comme telle dans la bdd. Toutefois, dans certains cas, il peut être utile de tout recalculer comme si la Bdd était entièrement non plane, puis réaffecter au cas par cas le caractère plat à certaines facettes.

Le bouton « ***Recalculer toutes les normales*** », comme son nom l’indique, recalcule toutes les normales de la bdd, c'est-à-dire les normales aux barycentres des facettes et les normales aux sommets. Le comportement pour les facettes préalablement déclarées planes est dépendant de la case à cocher située juste au-dessus. Pour ces dernières, le calcul fait ici est ne tient pas encore compte des angles de seuil entrés dans le menu Fichier/Préférences. Il utilise toutes les facettes qui partagent un sommet suivant l’algorithme décrit en Annexe B - . Les seuils définis dans le menu Fichier/Préférences ne seront appliqués qu’au moment de la visualisation et/ou de l’enregistrement du fichier.

Le bouton « ***Simplifier la BDD*** » s’utilise lorsqu’une base de données possède des éléments en double ou des éléments non utilisés. Il s’utilise également pour joindre des points d’un objet dont la distance est inférieure à la tolérance d’égalité décrite ci-dessus. En particulier, on trouve des bases de données dont les sommets des facettes sont déclarés comme différents bien qu’ils soient dans le même objet mais possèdent pourtant exactement les mêmes coordonnées x, y et z. Les points étant déclarés différents, les normales aux sommets sont elles aussi différentes et le lissage de Gouraud peut ne pas fonctionner, surtout après un recalcul des normales. La simplification de la BDD éliminera ce problème. Le processus est récursif, plutôt complexe et peut être long en temps d’exécution. Si deux points d’un même objet sont identiques (mêmes x, y et z à la tolérance près), il faut alors remplacer ce point dans toutes les facettes qui l’utilisent, éliminer ensuite les éventuels doublons à l’intérieur même d’une facette (cas où on avait dans une même facette 2 points très proches de distance inférieure à la tolérance), vérifier que les facettes ne sont pas dégénérées (restent à 3 points et plus), modifier les normales en conséquence, mettre à jour le tableau des arêtes, etc. Ce calcul peut énormément modifier la structure des tableaux en mémoire, leurs dimensions. Le tout est réalisé par une gestion dynamique de la mémoire qui n’est pas sans avoir posé des problèmes de mise au point, et a, par le passé été la cause d’un certain nombre de plantages d’OVNI en version Tcl/Tk. Il n’est hélas pas exclu que dans certains cas particuliers on ne retombe sur ce type de problème, même en version wxWidgets car la gestion de la mémoire étant très différente, l’algorithme a été complètement réécrit. Par prudence, avant de lancer une simplification de BDD, enregistrer la base de données avant simplification, surtout si on y a apporté des modifications. L’option de sauvegarde automatique, activée par défaut, peut venir en aide, mais on ne sait jamais.

Le bouton « **Recalculer les arêtes** » sert à forcer un nouveau calcul des arêtes de facettes. Le calcul de base peut être assez long quand on cherche à éliminer de façon récursive les arêtes en double. En effet, dans le calcul simple, une arête peut être utilisée par au moins 2 facettes adjacentes, ce qui pose des soucis lorsqu’on souhaite subdiviser des arêtes. Lorsque le calcul des doublons d’arêtes est trop long pour un objet, typiquement 2 secondes, on évite de le faire si le besoin n’est pas immédiat (comme par exemple dans la fonction de soudure entre points, où un Undo). Dans ce cas, l’utilisateur peut forcer le calcul manuellement en cliquant sur ce bouton.

La case à cocher « ***Activer la transparence*** » rend semi transparentes les facettes. Elle peut être utile pour visualiser plus facilement des objets cachés, notamment ceux inclus à l’intérieur d’autres. Elle peut aussi permettre de sélectionner des points qui sinon seraient cachés par une surface placée entre l’observateur et ce point. À utiliser cependant avec modération car il devient difficile parfois de sélectionner exactement le point ou la facette voulue.

#### Le menu Transformations

|  |
| --- |
|  |

Les transformations indiquées dans ce menu sont appliquées directement sur toute la base de données. La visualisation est mise à jour au fur et à mesure.

Les trois premières permettent de changer le signe des coordonnées sur chacun des 3 axes.

Les trois suivantes permutent les coordonnées des points sur 2 des 3 axes.

La suivante effectue une permutation circulaire sur les 3 axes des coordonnées des points.

Le sous-menu « ***Inverser toutes les normales*** » change le sens des normales de toutes les facettes, c’est à dire les normales au barycentre et les normales aux sommets. Le sous-menu « ***Inverser les normales* Sélectionnées** » n’applique ce changement qu’aux facettes sélectionnées.

|  |
| --- |
|  |

Le sous-menu « ***Déplacer la bdd*** » ouvre une boîte de dialogue dans laquelle ou pourra entrer les déplacements en x, y et z de l’ensemble de la base de donnée. Les déplacements entrés dans les fenêtres doivent être validés par le bouton « Accepter ». On peut inverser l’effet précédemment introduit par le bouton « Inverser ». On peut également centrer automatiquement la bdd sur le centre de la boîte englobante par le bouton « Centrage auto sur Min-Max de la bdd ». Le bouton quitter permet de fermer de cette fenêtre. Si les actions n’ont pas été validées, soit par le bouton « Accepter » soit par le bouton « Inverser », le bouton « Quitter » ne change pas la bdd. ***Note :*** *il arrive que la bdd ne s’affiche pas correctement, notamment lorsque son centre géométrique est trop éloigné du centre de rotation (souci avec zNear et Zfar dans OpenGL). Le centrage auto permet de contrecarrer cet effet indésirable.*

Le sous-menu « ***Facteur d’échelle*** » ouvre une boîte de dialogue dans laquelle ou pourra entrer un facteur de zoom à appliquer à l’ensemble de la base de donnée. Il possède les mêmes fonctionnalités que la boîte précédente avec les 3 boutons « Accepter », « Inverser » et « Quitter ». Un facteur de 2 multiplie toutes les coordonnées par 2. L’effet n’est toutefois pas directement visible dans la fenêtre OpenGL en raison du réajustement automatique du facteur de zoom visuel aux dimensions de la boîte englobante. On peut le vérifier via les propriétés de la Bdd (via le menu Fichier).

À noter que les actions de ces deux derniers sous-menus sont différentes de celles trouvées via le menu Outils/Sélection et déplacements/Manipulations de l’objet. Ici on agit toujours directement sur les coordonnées des points de toute la base de données. Par contre, via le menu Outil, on n’agit que sur un ou plusieurs objets, mais pas forcément tous, et de plus, l’action est faite en 2 temps : elle est d’abord purement graphique via les fonctions OpenGL adéquates et ce n’est qu’en validant par le bouton « Appliquer » que la base de donnée est modifiée.

#### Le menu Options

|  |
| --- |
|  |

On trouve dans ce menu des options ou fonctions générales, ne dépendant pas d’une base de données spécifique et qui ne trouvent pas leur place dans les autres menus plus spécialisés.

Les trois premiers sous menus permettent de retravailler les palettes de couleurs destinées à afficher et coloriser les groupes au sens « Crira ».

Le premier affiche une boîte de dialogue où on va pouvoir entrer un numéro de groupe et/ou un numéro de matériau. Pour chacun, à l’aide d’une palette de couleur, on va pouvoir choisir les composantes de lumière diffuse et de lumière ambiante (correspondant respectivement aux mots clés GL\_DIFFUSE et GL\_AMBIENT des routines OpenGL comme glMaterialfv). Les couleurs proposées en entrée sont celles de la palette en cours. Dans l’idéal, il faudrait aussi proposer des options pour modifier deux autres composantes, la part spéculaire (correspondant à GL\_SPECULAR) et la brillance (correspondant à GL\_SHININESS).

Les deux sous-menus suivants permettent soit de relire une palette existante (fichier texte avec l’extension .pal) soit d’enregistrer une palette sous un nouveau nom.

Par défaut, OVNI possède une palette intégrée au code et qu’on retrouve dans le fichier default.pal dans le répertoire de l’exécutable. Une seconde palette, b747 y est également disponible à titre de démonstration.

OVNI prévoit de manipuler jusqu’à 32 couleurs de groupes (suffisant pour Crira qui en version 6.4 n’en a que 10).

Dans une prochaine version d’OVNI, il pourrait être intéressant d’offrir d’avantage d’options de manipulation de ces palettes de couleur en y ajoutant également la modification du fond bleu par défaut et du gris standard, sans colorisation de groupe.

Le sous-menu « Zoom automatique » peut s’utiliser si on veut revenir à une valeur de Zoom permettant un affichage de l’ensemble des objets, surtout après diverses modifications de zoom mais également ajout, suppression d’objets, modification d’échelle, etc. La valeur du Zoom à appliquer est calculée à partir de la boîte englobante des objets, sans compter les objets « à supprimer ».

Le sous-menu « Centrage automatique » place le centre de la boite englobante des objets au centre de l’écran. Utile si on a déplacé plusieurs fois le centre de rotation ou perdu l’affichage des objets car la valeur de zoom est trop sensible aux déplacements latéraux.

#### Le menu Aide

Ce menu contient 2 sous-menus.

Le premier sous-menu (Aide), également accessible via la touche F1, appelle une fenêtre de visualisation de fichiers html gérés par wxWidgets. Ces fichiers sont placés dans un sous répertoire « help » du répertoire où se trouve l’exécutable d’OVNI.

Le point d’entrée est le fichier « Aide.html ». Ce fichier comporte un lien vers un deux autres fichier « Menu.html » et « About.html ».

Ces fichiers contiennent un rappel abrégé des diverses commandes, menus et sous-menus d’OVNI.

|  |
| --- |
|  |

Le second sous-menu (A propos) affiche une information sur Ovni, donnant, notamment le numéro et le type de version de la librairie wxWidgets utilisée ainsi que le compilateur et le nombre de bits de compilation.

### Les boutons sous la barre de menus

La barre de boutons se présente ainsi :



Les boutons + et – permettent d’augmenter ou de diminuer la valeur du Zoom sur tous les objets affichés. Il existe un raccourci clavier faisant la même chose.

Les boutons de déplacements gauche, droite, haut, bas, déplacent à l’écran tous les objets affichés dans le sens indiqué. Il existe des raccourcis clavier faisant la même chose (flèches du clavier).

Les boutons à icône qui suivent permettent d’accéder rapidement à certaines fonctionnalités, pour la plupart présentes via les divers menus. Pour une explication plus détaillée, on se reportera donc au menu correspondant.

* points : active/désactive l’affichage en mode points (voir aussi §5.1.3.2) ;
* filaire : active/désactive l’affichage en mode filaire ;
* plein : active/désactive l’affichage en mode plein (activé par défaut) ;
* axes : active/désactive l’affichage des 3 axes X, Y et Z de la base de données (voir aussi §5.1.3.2) ;
* boite : active/désactive l’affichage de la boîte englobante des objets affichés ;
* B_Normales : active/désactive l’affichage des normales au barycentre des facettes sélectionnées (voir aussi §5.1.3.4) ;
* S_Normales : active/désactive l’affichage des normales aux sommets des facettes sélectionnées ;
* sun : active/désactive l’affichage de la source d’éclairage OpenGL ;
* gouraud : active/désactive le lissage (l’ombrage) de Gouraud ;
* outils : affiche la boîte de dialogue Outils/Sélection et déplacements (voir §5.1.3.7.1) ;
* modifs : affiche la boîte de dialogue Outils/Modifications (voir §5.1.3.7.2) ;
* wxSlider16 : active/désactive l’affichage des curseurs en bas de fenêtre (activé par défaut).
*  : active/désactive l’affichage des groupes au sens Crira (désactivé par défaut).
*  : active/désactive l’affichage des matériaux (désactivé par défaut).

Les boutons ont deux états : enfoncé pour activer la fonction correspondante ou relâché pour désactiver la fonction.

Si certaines des fonctions peuvent également être activées par un raccourci au clavier (voir ci-dessous §5.2), ou via un bouton, une case à cocher dans un menu et/ou une boîte de dialogue, l’état des boutons change en fonction de l’état courant de la fonction.

## Les raccourcis au clavier

Voir aussi le paragraphe 5.1.2 Mouvements de base au clavier.

Les ***flèches du clavier*** (droite, gauche, haut et bas) sont utilisées pour les translations en X et Y (écran).

Pour obtenir une rotation en X ou Y écran, appuyer simultanément sur la touche ***Shift*** (ou ***Maj.***) et les flèches du clavier.

Les flèches du clavier numériques (Keypad) sont aussi utilisées pour les rotations ou des déplacements en X et Y écran (touches ***6***, ***4***, ***8*** et ***2***, associées ou non à la touche ***Maj***).

Sur PC, l’action obtenue dépend à la fois de l’état de la touche de verrouillage du pavé numérique ("Num Lock" ou "Verr num") et de la pression conjointe sur la touche **Maj**.

Les touches ***+*** et ***-*** du clavier numérique peuvent aussi être utilisées pour activer le Zoom.

Les touches ***Page up*** et ***Page down*** ont la même action mais en Zoom rapide (x20).

La touche **c** centre la rotation soit sur le barycentre des objets, soit sur le barycentre de la sélection, quel que soit le mode de sélection, Facettes par défaut, mais aussi objets ou points.

La touche **C** (en majuscule) est l’équivalent du bouton d’activation / désactivation de l’affichage avec ou sans les curseurs (ou sliders).

La touche **g** (ou **G**) effectue une permutation circulaire sur les différents modes de colorisation des facettes : par groupes, matériaux ou sans colorisation.

La touche **m** affiche dans la fenêtre Dos les caractéristiques de la matrice de rotation.

La touche **M** (en majuscule) est l’équivalent du bouton d’activation / désactivation du mode d’affichage avec ou lissage de Gouraud (GL\_FLAT et GL\_SMOOTH). Le mode smooth applique un lissage de Gouraud nécessitant la présence de normales aux sommets. ATTENTION à la valeur du seuil dans le menu "Préférences". Une valeur de 35° semble un bon compromis dans la plupart des cas. *Cette fonction est directement accessible via une icône de la barre de menus.*

Sur certaines cartes graphiques le mode SMOOTH pose problème (dégradés de couleurs à l'intérieur de facettes normalement uniformes). C’est par exemple le cas sur un portable possédant une puce graphique Intel 855 GM. Passer alors en mode Flat (*En version Tcl/Tk on pouvait aussi utiliser le driver Mesa (sous Windows), mais non testé avec wxWidgets*).

La touche **N** (on **n**) est l’équivalent du menu « Affichage / Sens des normales ».

La touche ***v*** (ou ***V***) permet de basculer entre 2 modes d’affichage des informations dans la console DOS : un mode standard, par défaut, et un mode bavard (v pour verbose) donnant d’avantage d’informations à chaque étape de calcul. Ce mode est surtout intéressant pour mieux comprendre ce qui se passe, mais plutôt à des fins de débogage.

La touche ***s*** (ou ***S***) permet de remettre à zéro la sélection en cours (facettes, points ou objets).

La touche **x** permet d’étendre une sélection de facettes. L’action de cette touche revient à comparer les normales aux barycentres des facettes sélectionnées avec les normales des facettes voisines. Lorsque les écarts entre les normales de la sélection originale et celle de la facette à tester sont inférieurs à 60° (valeur en interne du code : compare\_normales) on sélectionne automatiquement la nouvelle facette. L’extension se fait de proche en proche. L’avantage est ainsi d’augmenter rapidement une sélection sans cliquer sur chacune des facettes. Lorsque plus aucune facette jointive à la sélection en cours ne correspond au critère, un message d’avertissement est affiché. Il est bien entendu toujours possible de sélectionner une facette supplémentaire via un clic sur le bouton central (ou la molette) mais aussi de désélectionner une facette que l’utilisateur juge en trop dans la sélection automatique obtenue.

La touche **i** (ou **I**) permet d’inverser les normales des facettes sélectionnées. Si aucune facette n’est sélectionnée, cette touche est équivalente à inverser toutes les normales.

La touche **f** (ou **F**) peut s’utiliser pour invalider un test limitant la recherche de doublons d’arêtes et leur simplification. Par défaut, si le nombre d’arêtes est supérieur à 200000, on ne fait pas cette recherche car elle devient vite très longue. Elle n’a de réel intérêt que lorsqu’on ajoute des sommets sur des arêtes. On peut court-circuiter cette limite via la touche **F**, mais attention car le calcul peut vraiment devenir très long.

La touche **p** (ou **P**) permet d’inverser le sens de parcours des facettes sélectionnées.

Par défaut, les facettes qui ont été sélectionnées le restent après l’utilisation des 2 options ci-dessus. Utile en cas d’erreur, mais peut devenir gênant à l’usage quand on doit utiliser plusieurs fois ces options de façon consécutives car il faut passer par l’option de la touche ‘s’ à chaque fois. La touche **z** (ou **Z**) permet d’inverser le comportement courant : en option on enchaine alors automatiquement l’option ‘s’, c’est-à-dire qu’en fin d’action, les facettes seront automatiquement désélectionnées. On peut réactiver le comportement par défaut en appuyant une seconde fois sur cette touche **z**.

La touche ***t*** (ou ***T***) est l'équivalent du menu Préférences/Mode de Rotation à la souris. Elle permet de basculer du mode de rotation directe des axes X, Y et Z de la bdd vers le mode "Trackball" (et réciproquement).

***Note 1*** : Dans le mode direct, les mouvements en X et Y de la souris n'agissent que sur les axes X et Z de la bdd. La seule façon d'obtenir une rotation autour de l'axe Y est d'activer le curseur Y en bas de l'écran.

Dans le mode "Trackball" la bdd tourne autour des axes X et Y de l'écran. On agit ainsi indirectement sur les 3 axes de la bdd en combinant les rotations en X et Y écran vis une matrice de rotation interne au code et dépendant de l’état des axes au début du mouvement et donc au moment du clic gauche sur le bouton de la souris.

***Note 2*** : Les curseurs agissent toujours directement sur les axes X, Y et Z de la bdd.

***Note 3*** : En mode "Trackball" les valeurs de rotation en Y ]90,270[ ne peuvent être atteintes par des mouvements à la souris bien que ce soit possible avec le curseur Y. C'est normal. Dans ce cas, les valeurs des angles en X et Z compensent cette zone "interdite" des angles en Y.

La touche ***r*** (ou ***R*** *pour Reset*) du clavier réinitialise aux valeurs par défaut les angles et les déplacements latéraux en x, y et z (clavier et/ou souris), ainsi que la valeur du zoom.

## Les menus contextuels

Cette possibilité est nouvelle en version wxWidgets. En version Tcl/Tk, elle n’avait pas été implémentée (mais aurait pu l’être).

En version wxWidgets, on peut sélectionner directement dans la fenêtre OpenGL, par défaut, des facettes via un clic sur le bouton central (ou la molette) de la souris. Ce n’était pas fait ainsi en version Tcl/Tk car il fallait passer systématiquement pas le menu (ou le bouton) « Sélections et Déplacements ».

|  |
| --- |
|  |

Un clic droit n’importe où dans la fenêtre OpenGL fait apparaître le menu contextuel suivant. On peut ainsi remettre à zéro la sélection de facettes, centrer sur la sélection, étendre la sélection, masquer ou supprimer les facettes sélectionnées, inverser les normales ou seulement le sens de parcours des facettes, afficher les normales aux barycentres ou aux sommets. La plupart de ces fonctions sont aussi disponibles via un raccourci clavier rappelé entre parenthèses (et décrits au §5.2), à l’extrémité droite des menus contextuels.

Les fonctions en grisé ne sont pas disponibles dans le contexte actuel, mais le deviendront dès qu’un changement de contexte autorisera leur fonctionnalité. Dans l’exemple ci-dessus, le sous-menu « étendre la sélection » est désactivé. Il deviendra actif dès qu’au moins une facette aura été sélectionnée.

Via le menu ou le bouton « Sélections et Déplacements » on peut passer en mode sélections d’objets ou sélections de points. Les intitulés des sous-menus contextuels et leur mode activé/désactivé s’adaptent en conséquence.

# Installation

Sous Windows, une installation basique est très simple. Il suffit de dézipper le fichier fourni dans un répertoire de l’utilisateur en respectant la structure des sous-répertoires inclus dans ce fichier compressé. Tous les outils nécessaires sont fournis.

Sous Windows, ou peut également générer OVNI à partir des codes sources. Il suffit de suivre la procédure indiquée au §7.

## Exécutable Windows

Le fichier compressé contient typiquement les éléments suivants :

Un répertoire de base : Ovni\_wxWidgets que l’utilisateur copie où il veut dans son espace de travail. Il peut aussi renommer à sa guise ce nom de répertoire.

Ovni\_wxWidgets contient en standard un sous-répertoire (***bin*** *ou* ***x64***) ainsi qu’en général 2 fichiers (Bugs.doc et 00\_Attention.txt). Les 2 fichiers peuvent ne pas être distribués systématiquement.

Le sous-répertoire ***bin*** ou ***x64***, contient :

* la version exécutable d’OVNI (par exemple Ovni\_64.exe);
* des fichiers d’exemples de bases de données au format SDM Oktal (extension .bdd), un ou des fichiers Autodesk (extension .3ds), un ou des fichiers Wavefront Obj (extension .obj), un ou des fichiers G3D (extension .g3d), un ou des fichiers Stéréolithographiques (extension .stl)  ;
* des fichiers de palettes de couleurs (default.pal et b747.pal) ;
* le fichier freeglut.dll (version libre et actuelle de la librairie glut qui n’est plus maintenue) ;
* les fichiers wxmsw32u\_gcc\_cb\_64.dll et wxmsw32u\_gl\_gcc\_cb\_64.dll qui contiennent respectivement les librairies wxWidgets 3.2.0 et la partie OpenGL de wxWidgets 3.2.0, compilées en 64 bits.
* un certain nombre de fichiers texte (extension .txt) donnant quelques informations complémentaires
* un sous répertoire help contenant des fichiers d’aide au format html ;
* un sous répertoire Icones contenant plusieurs icônes utilisées dans l’interface d’OVNI.
* Quelques dll complémentaires, indispensables avec le compilateur gcc 8.1 et au-dessus : libgcc\_s\_seh-1.dll, libstdc++-6.dll, libwinpthread-1.dll. *Note* : si le compilateur gcc 8.1, 64 bits, version seh-posix est installé par défaut sur la machine Windows et est accessible via la variable path du système, ces 3 dll ne sont pas utiles car déjà vues.

### Remarques 1

Avec certaines cartes graphiques, et avec les versions Tcl/Tk, on pouvait avoir des soucis de présentation des couleurs dans les facettes sous OpenGL : par exemple une facette grise, bleu, rouge,… devant être colorisée de façon homogène pouvait présenter des dégradés de couleurs. C’est un problème d’interface entre certaines librairies du système et le hardware graphique de la carte. C’est un problème connu, qui n’est pas spécifique à OVNI. Cela se produisait par exemple sur un portable possédant une puce graphique Intel 855 GM, plutôt bas de gamme. La solution consistait à utiliser des versions dites « Mesa » de certaines librairies, qui court-circuitent alors une partie du hardware non compatible de la carte. En version wxWidgets, on n’est pas tombé sur ce type de soucis, mais le nombre de machines testées est trop faible pour identifier un tel problème. Les librairies incriminées étaient GLU32.DLL et OPENGL32.DLL, qui devaient impérativement être dans le même répertoire que l’exécutable. De ce fait, ce sont ces versions qui étaient utilisées par OVNI et non celles existant déjà dans le système.

Si cela se produit, on pourrait tenter de réinstaller ces fichiers Mesa, mais il faudra alors faire attention à la façon dont ils ont été compilés. Si Ovni est en 32 bits, les dll devront être de même : c’est le cas des versions Tcl/Tk d’Ovni. Si Ovni est compilé en 64 bits, il faudra trouver des versions 64 bits, Mesa des dlls, ce qui n’a pas été fait ici.

### Remarques 2

La façon la plus simple d’utiliser OVNI sous Windows est d’associer les extensions .bdd, .3ds, .obj, .off, .m3d, .ply, .g3d et .stl avec OVNI. Ainsi, on lancera automatiquement OVNI par double clic sur un fichier de base de données. Il se peut que certaines extensions soient déjà utilisées par ailleurs (notamment .obj, fichier issu d’une compilation par Visual C). Dans ce cas, soit on force l’association avec OVNI soit on passe par le menu « Ouvrir » d’OVNI. Le fichier Note\_3.9.txt, intégré au fichier zip, donne des méthodes d’association pour Windows 2000 et Windows XP ainsi que la façon d’associer une icône avec ces fichiers. Sous Windows 7 ou 10, les procédures diffèrent un peu mais restent assez semblables. Toutefois, il arrive que ces méthodes ne donnent pas le résultat escompté car on n’a pas les droits suffisants ou que Windows refuse de le faire pour une raison quelconque. Il faudra alors demander à un administrateur ou utiliser d’autres façons de faire qui sortent du cadre de ce document. Consulter, par exemple, « Windows pour les Nuls » !!!

## Librairies diverses

Les librairies citées ici intéressent plutôt les développeurs qui devront choisir entre les différentes versions possibles, 32/64 bits, statiques/dynamiques.

### OpenGL

Pas de soucis particuliers avec OpenGL normalement. Il est supporté nativement par les operating system divers, et donc Windows. En cas de problèmes avec le hardware graphique, il pourrait s’avérer utile de faire installer des versions Mesa des drivers (notamment opengl32 et glu32). Une partie des fonctions normalement confiées au hardware de la carte graphique sera alors prise en charge par du software au détriment des performances.

### Glut

La librairie Glut est une surcouche d’OpenGL offrant des fonctions avancées de gestion et des routines de plus haut niveau que l’OpenGL de base. OVNI a été développé initialement à partir de la version Glut 3.7, disponible à l’époque sur Sun Solaris, Linux, Windows. Cette version est maintenant obsolète. On trouve sous Linux et Windows une autre distribution dénommée FreeGlut, compatible du Glut original et qui a été testé avec succès sur nos machines. Cette librairie comporte deux parties, une librairie dynamique proprement dit (un fichier freeglut.dll ici, disponible en 32 ou en 64 bits) et un ensemble de fichiers d’entête .h qui trouvent normalement leur place dans un sous-répertoire comme /include/GL du compilateur. A noter toutefois, qu’avec la dernière version de freeglut (3.0), un souci de plantage à l’exécution est apparu avec la fonction glutSolidSphere, utilisée pour tracer un cercle plein matérialisant la position de la source de lumière (petit cercle en jaune). Faute de mieux, on a intégré le code C de la version précédente (2.8.1) directement dans Ovni, ce qui a résolu le problème.

### Lib3ds

C’est une librairie et des entêtes (fichiers .h) utilisés pour décoder les fichiers de base de données au format Autodesk (extension .3ds).

Sous Windows, la librairie statique (fichier lib3ds-2\_0.a dans lib\_loc\lib3ds en 32 bits ou lib64\ lib3ds-2\_0\_64.a en 64 bits). On pourrait aussi utiliser une version dynamique de cette librairie.

On peut télécharger le code source de la librairie depuis le site <http://code.google.com/p/lib3ds>. C’est la version 1.3 qui était utilisée jusqu’à la version 3.29 d’OVNI Tcl/Tk. Depuis la version 3.30, on utilise la lib3ds version 2.0. On trouve sur ce site ce qu’il faut pour compiler la librairie, notamment des « makefile » de génération.

### Expat

C’est une librairie et des entêtes (fichiers .h) utilisés pour décoder et analyser les fichiers de base de données au format xml du groupe G3D (extension .g3d). À noter que cette librairie est bien connue dans tous les environnements : elle est utilisée notamment par le navigateur Firefox.

Sous Windows, la librairie statique (fichier libexpat-2\_0.a en 32 bits dans lib\_loc ou lib64\libwxexpat.a en 64 bits) et les fichiers d’entête sont fournis dans le fichier zip source.

On peut aussi télécharger le code source d’Expat depuis le site <http://expat.sourceforge.net> via la rubrique Downloads ou plus directement via <http://sourceforge.net/projects/expat>. C’est la version 2.0.1 qui est utilisée actuellement dans OVNI. On trouve sur ce site ce qu’il faut pour compiler la librairie, notamment des « makefile » de génération.

Dans notre cas, en version wxWidgets, nous avons utilisé la version intégrée à la génération de wxWidgets, soit en 32 bits, soit en 64 bits. Le nom de la librairie est légèrement différent (libwexpat.a) mais ça ne pose pas de problème particulier.

# Compilation d’OVNI

Le code source est fourni dans un autre fichier .zip que celui décrit ci-dessus pour l’installation simple. L’environnement de développement utilisé ici est Code::Blocks. L’intérêt de cet environnement dans notre cas, est d’intégrer wxSmith qui permet de construire assez facilement une interface graphique. De plus, Code::Blocks est multiplateforme, contrairement à la suite Visual réservée à Windows, ce qui permet d’obtenir quasiment le même environnement visuel graphique dans plusieurs operating system : Windows, Linux, OSX,.... Il existe d’autres environnements de développement libres comme Eclipse associé à wxFormBuilder pour aboutir au même résultat, mais c’est Code::Blocks que préfère l’auteur de ces lignes ! C’est donc dans cet environnement que seront donnés les fichiers projets de génération.

Code::Blocks est un environnement de développement qui peut s’adapter à de nombreux compilateurs. Le choix retenu ici est le portage MinGW en version 32 ou 64 bits des compilateurs standard de Linux. Code::Blocks est actuellement fourni avec une version 5.1 de gcc, version dite TDM. Cette version est un peu ancienne, basée sur la version 5.1 officielle mais avec quelques ajouts bien utiles. L’auteur en a toutefois arrêté le développement, en particulier à cause de difficultés croissantes au fur et à mesure de la publication des versions officielles, difficultés de maintenance de la compatibilité et liées à ses propres ajouts. De plus, le compilateur fortran, gfortran 5.1 (même officiel) est bugué, bien que cela n’ait pas d’impact pour Ovni. Sur le site <https://sourceforge.net/projects/mingw-w64/>, on trouve des versions récentes et parfaitement adaptées à leur utilisation dans Code::Blocks. Notre choix se porte sur la série MinGW-W64 GCC-8.1.0 et plus particulièrement les versions [x86\_64-posix-seh](https://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win64/Personal%20Builds/mingw-builds/8.1.0/threads-posix/seh/x86_64-8.1.0-release-posix-seh-rt_v6-rev0.7z) pour les compilateurs 64 bits et [i686-posix-sjlj](https://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win32/Personal%20Builds/mingw-builds/8.1.0/threads-posix/sjlj/i686-8.1.0-release-posix-sjlj-rt_v6-rev0.7z) pour les compilateurs 32 bits. D’autres choix sont possibles, et conviendraient tout autant, mais c’est, à notre connaissance et au moins pour la version 64 bits, ce qui sera fourni en standard avec les prochaines versions officielles de Code::Blocks (et est déjà utilisé pour les livraisons d’exécutables de l’environnement en versions « nightlies »). *Note* : la version des compilateurs 64 bits peut tout à fait compiler du code en 32 bits. Nous avons toutefois préféré garder *pour l’instant* les deux versions, installées dans des répertoires séparés, MinGW64 et MinGW32 respectivement. Dans la configuration de Code::Blocks, on choisira l’un ou l’autre en fonction du type d’exécutable désiré en sortie. Pour Ovni, ce sera du code en 64 bits de préférence. *Note* : en version Tcl/Tk, on ne pouvait générer que du code en 32 bits en raison des librairies statiques et/ou dynamiques disponibles. Avec wxWidgets, générer en 32 bits fonctionne très bien également. Toutefois, on est tombé sur au moins un cas à problèmes, sur une très grosse base de données comportant un peu moins d’un million de facettes, problème qui a disparu en 64 bits ! L’inconvénient, est qu’il faut trouver des versions 64 bits des librairies utilisées. Mais de nos jours, ce n’est plus vraiment un souci, d’autant plus que dans la plupart des cas, on est en mesure de re-générer complètement les dites librairies avec le (les) compilateur(s) installé(s) sur la machine cible.

Préambule :

Pour compiler Ovni, il faut avoir au préalable installé une version complète de wxWidgets. La seule installation des dlls ne suffit pas car il y manque tous les headers utilisés à la compilation des appels aux fonctions wxWidgets. Sur le site officiel wxWidgets (<https://www.wxwidgets.org/downloads/>) on peut trouver tout ce qu’il faut, y compris des liens vers des versions compilées des librairies et ce, pour divers compilateurs. Ces librairies, peuvent fonctionner en l’état, mais il est tout de même impératif d’avoir une installation complète des headers. Du coup, cette installation fournit aussi divers fichiers (ou makefile) de génération qu’on peut utiliser : on est ainsi certain que les dlls générées utiliseront le même compilateur que pour Ovni (voir **Erreur ! Source du renvoi introuvable.Erreur ! Source du renvoi introuvable.**). On peut aussi trouver ces librairies pour Windows sur <http://www.wxdev.fr/> et plus particulièrement sur <https://wxstuff.xaviou.fr/wxlibs/wx312.html>, site maintenu par un français et relatif à diverses utilisations des wxWidgets. Le site comporte de nombreux tutoriels.

Comme dit en début de paragraphe, le développement d’Ovni a été réalisé sous l’environnement de développement Code::Blocks. Les versions officielles (dites releases) se trouvent sur <http://www.codeblocks.org/>. La dernière en date est une version 17.12 (décembre 2017). Une autre devrait sortir en 2019, mais se fait attendre. Il est possible d’installer « par-dessus » la version 17.12, une version dite « nightly », plus récente et, en général, stable. Voir la page : <http://forums.codeblocks.org/index.php/board,20.0.html>, etpar exemple, la page <http://forums.codeblocks.org/index.php/topic,23234.0.html>, qui est celle de la version du 20 avril 2019 : bien suivre les diverses explications. Comme dit un peu plus haut, la version 17.12 est fournie avec gcc 5.1. On a tout intérêt à remplacer ce compilateur par une version plus récente, la 8.1 en l’état actuel. C’est d’ailleurs avec ce compilateur que les dernières « nightlies » ainsi que les dlls wxWidgets 3.14 utilisées par Code::Blocks ont été générées.

***Note*** : On trouve d’autres versions de compilateur MinGW64 sur internet. Les derniers développements d’Ovni sont faits à partir de l’ensemble d’outils Msys2. L’avantage est que ce système est mis à jour régulièrement, en est en version 12.1 et offre un utilitaire de mise à jour automatisé (pacman).

**Msys2**

Home page : <http://www.msys2.org/>

pacman -Syu

voir aussi : <https://solarianprogrammer.com/2019/11/05/install-gcc-windows/>

Packages de base : <https://packages.msys2.org/base>

Ou via :

<https://packages.msys2.org/group/mingw-w64-x86_64-toolchain>

pacman -S mingw-w64-x86\_64-binutils

pacman -S mingw-w64-x86\_64-crt-git

pacman -S mingw-w64-x86\_64-gcc

pacman -S mingw-w64-x86\_64-gcc-fortran

pacman -S mingw-w64-x86\_64-gcc-libgfortran

pacman -S mingw-w64-x86\_64-gcc-libs

pacman -S mingw-w64-x86\_64-gdb

pacman -S mingw-w64-x86\_64-headers-git

pacman -S mingw-w64-x86\_64-libmangle-git

pacman -S mingw-w64-x86\_64-make

pacman -S mingw-w64-x86\_64-pkg-config

pacman -S mingw-w64-x86\_64-tools-git

pacman -S mingw-w64-x86\_64-libwinpthread-git

pacman -S mingw-w64-x86\_64-winpthreads-git

pacman -S mingw-w64-x86\_64-winstorecompat-git

Note :

pacman -S mingw-w64-x86\_64-toolchain

doit tout faire.

On peut installer ces packages dans C:\msys64. On y trouve alors msys2.exe qui lance une fenêtre de commande de type DOS dans laquelle on entre les diverses commandes comme pacman, … Plusieurs sous-répertoires seront créés et notamment mingw64 contenant les outils et compilateurs 64 bits. On y trouvera aussi un sous-répertoire mingw32, clang32, clang64 prévus pour y installer d’autres compilateurs sous Msys2.

Il faudra ensuite ajuster la variable path pour que Windows trouve facilement les utilitaires contenus dans C :\msys64\mingw64\bin.

1. Format des fichiers 3D
2. Généralités

Les différents formats de fichiers listés ici comprennent au minimum la notion de points et la notion de facette. Les points sont entrés sous la forme des coordonnées x, y, z dans l’espace. Ces points sont rassemblés dans un ou plusieurs tableaux de points. Les facettes sont composées de plusieurs points, au moins 3. En général, les points utilisés par une facette sont repérés par un numéro d’indice dans le tableau des points correspondants.

La plupart des formats connaissent la notion d’objet. C’est un ensemble de facettes groupées, généralement par fonctionnalité : une aile d’avion, une tuyère, une roue, …

La plupart des formats connaissent la notion de normale à la facette. Deux types de normales existent : celle au barycentre de la facette et celles aux sommets de la facette. Dans le premier cas, il n’y a qu’une seule normale par facette, même si la facette comporte plus de trois points pas forcément coplanaires. La normale utilisée est alors plus ou moins une direction normale moyenne. Dans le second cas, il y a autant de normales que de sommets. Ces normales sont utilisées pour lisser en apparence les facettes jointives : le lissage de Gouraud notamment utilise ces normales pour calculer une apparence lissée des facettes par interpolation bilinéaire. Ce lissage est natif dans OpenGL et utilise directement les possibilités hardware de la carte graphique. Si deux facettes jointives représentent effectivement une surface continue, sans angles, les normales aux sommets de ces deux facettes sont communes. Si par contre, les deux facettes doivent présenter un angle entre elles, on pourra avoir plusieurs normales par sommet, au maximum autant de normales que de facettes partageant ce sommet.

Le lissage de Gouraud convient bien à la représentation des objets dans OVNI. En chaque sommet d’une facette, à l’aide de la normale au sommet, OpenGL calcule une couleur, ou dans notre cas un niveau de gris, en fonction de la normale au sommet, des conditions d’éclairage liées aux positions de la source et de l’observateur, de la couleur de la facette. La couleur en chaque pixel interne d’une facette est calculée par interpolation bilinéaire sur la couleur de tous ces sommets. Les cartes graphiques modernes implantent dans leur hardware cette fonction. Dans Crira, on va faire un peu différemment et utiliser plutôt un lissage de Phong. Là aussi, on a besoin des normales aux sommets, mais on va d’abord calculer en chaque pixel interne d’une facette une normale par interpolation bilinéaire et c’est seulement ensuite qu’on calcule la « couleur » ou le niveau de gris du pixel. L’algorithme de Phong demande davantage de calcul que celui de Gouraud, mais est plus proche de la vérité. Notons cependant que même si la normale varie d’un pixel à son voisin, la surface d’une facette reste plane, ce qui ne remplace pas complètement une description fine des surfaces.

En général, mais pas forcément, les objets sont indépendants les uns des autres : chacun a son tableau de points, son tableau de facettes, ses tableaux de normales. Pour OVNI, basé sur l’utilisation du format SDM Oktal-SE, c’est la règle. Il n’y aura donc pas formellement de points communs entre 2 objets.

OVNI ne supporte pas forcément toutes les options de chacun des formats. Par exemple, les couleurs, les textures, sont ignorées. Pour chacun des formats ci-dessous, on indiquera quelles sont les options qui sont reconnues. Dans l’ensemble, une option inconnue est ignorée. Sur l’ensemble des fichiers testés jusqu’à présent, il n’y a que 2 ou 3 cas particuliers qui posent problème. Il suffit alors de lire ce fichier avec un logiciel du commerce comme Deep Exploration (version 6.2 en 2010) et de réenregistrer le fichier dans le même format. En général, le problème est alors résolu car la fonction un peu spéciale n’est pas reconduite lors de l’enregistrement, quitte à accroître la taille du fichier.

1. Fichiers SDM Oktal-SE (.bdd)

C’est un format de type texte piloté par des mots clés ou balises. La portée d’une balise est limitée par l’entrée d’une autre balise. Il n’y a pas dans ce format, comme sous xml, un couple comme <face> … </face>.

En général, la fin des données liées à une balise est séparée de la balise suivante par une ligne blanche, mais ce n’est pas obligatoire. Lorsqu’une ligne comporte plusieurs données, celles-ci sont séparées par au moins un espace blanc et/ou une tabulation.

OVNI ne supporte qu’un sous-ensemble des mots clés du format SDM Oktal-SE. Le format prévoit que les mots clés soient en français ou en anglais. OVNI reconnaît indifféremment les deux types, du moins pour ceux qui sont supportés.

Ce format est multi objet. Chaque objet possède au moins un tableau de points et un tableau de facettes. Il peut avoir en option un tableau de normales aux facettes, un tableau de normales aux sommets, un tableau de numéros de groupes, un tableau de numéros de matériaux et éventuellement une matrice de transformation associée à un objet. Cette matrice sert à modifier l’échelle de l’objet, à le déplacer ou à en effectuer une rotation. La matrice est lue et traitée en entrée dans OVNI. Elle n’est pas recréée en sortie car ce sont directement les points modifiés par le traitement matriciel qui sont enregistrés.

L’ordre d’apparition des mots clés à l’intérieur d’un objet n’est pas figé. Néanmoins, il est préférable d’utiliser l’ordre ci-dessous afin de faciliter certains contrôles de validité interne à OVNI.

Un fichier typique sur un cube est donné ci-dessous :

#######################################################

<OBJET> 1 Cube

<FACE> 6

1 4 1 2 3 4

2 4 1 4 6 5

3 4 5 6 8 7

4 4 7 8 3 2

5 4 1 5 7 2

6 4 3 8 6 4

<SOMMET> 8

1 0.0 1.000 1.000

2 0.0 -1.000 1.000

3 0.0 -1.000 -1.000

4 0.0 1.000 -1.000

5 -2.00 1.000 1.000

6 -2.00 1.000 -1.000

7 -2.00 -1.000 1.000

8 -2.00 -1.000 -1.000

<NORMALE> 6

1 1. 0. 0.

2 0. 1. 0.

3 -1. 0. 0.

4 0. -1. 0.

5 0. 0. 1.

6 0. 0. -1.

<ASPECT\_FACE> 6

1 <GROUPE> 2

2 <GROUPE> 2

3 <GROUPE> 2

4 <GROUPE> 2

5 <GROUPE> 2

6 <GROUPE> 2

<POSITION> 1

1 <MAT\_POSITION> 1.00000000 0.00000000 0.00000000 0.00000000

0.00000000 0.25000000 0.00000000 0.00000000

0.00000000 0.00000000 1.00000000 0.00000000

0.00000000 0.00000000 2.00000000 1.00000000

######################################################################

Les lignes commençant par un dièse sont considérées comme des commentaires. Elles sont ignorées par OVNI.

Le mot clé <OBJET> (ou <OBJECT> en version anglaise) est suivi, sur la même ligne, d’un numéro d’objet et d’un nom d’objet. ***Attention si ce nom comporte des caractères blancs ou des caractères accentués***. En général, OVNI s’en accommode, du moins si l’encodage du fichier est en utf-8 (sans BOM), mais ce n’est pas forcément le cas par ailleurs, notamment dans Crira. OVNI supporte et détecte en entrée ligne par ligne, l’encodage utf-8 ou Ansi. Pour Ovni, ce qui n’est pas utf-8 est considéré comme Ansi. En cas de doute, remplacer les caractères accentués par des caractères équivalents, sans accent. Chaque apparition d’une balise <OBJET> défini un nouvel objet. Bien que cela n’ait pas d’importance pour OVNI, il est préférable que deux objets n’aient pas le même numéro ni le même nom. En sortie d’OVNI, les fichiers créés seront d’office créés en encodage utf-8. En fait, tant qu’il n’y a pas de caractères accentués, les 2 encodages produisent exactement la même chose.

Le mot clé <SOMMET> (ou <VERTEX> en version anglaise) est suivi sur la même ligne du nombre de points qui forment l’objet (par exemple 8 ci-dessus).

Les lignes suivantes comportent d’abord le numéro de point (entier) puis les 3 coordonnées (format flottant) en x, y et z du point. Le format est libre, séparé par des espaces et/ou des tabulations.

Il doit y avoir autant de lignes que de nombre de points déclarés avec le mot-clé. La numérotation commence à 1. Il n’est pas obligatoire d’entrer les coordonnées dans l’ordre, mais c’est préférable. S’il manque une valeur, OVNI le signalera.

Le mot clé <FACE> (ou <POLYGON> en version anglaise) permet de définir les facettes d’un objet. Le mot clé est suivi du nombre de facettes (6 dans l’exemple ci-dessus).

Les lignes suivantes comportent d’abord le numéro de la facette, suivi du nombre de sommets de la facette et enfin, pour chaque sommet, le numéro d’indice des sommets dans le tableau des points. La numérotation des facettes commence à 1. Il doit y en avoir autant que de nombre déclaré sur la ligne du mot clé. S’il manque une valeur, OVNI le signalera.

Seuls les mots clés <OBJET>, <SOMMET> et <FACE> sont obligatoires. Le reste est optionnel en entrée d’OVNI.

Le mot clé <NORMALE> (ou <POLY\_NORMAL> en version anglaise) permet d’entrer les normales aux barycentres des facettes. Il est suivi du nombre de normales, en général égal aux nombres de facettes, mais ce n’est pas une obligation (6 dans l’exemple ci-dessus).

Les lignes suivantes contiennent d’abord un entier utilisé comme indice dans le tableau des normales. Le numéro entré correspond au numéro de facette à laquelle est affectée cette normale. Puis viennent les 3 coordonnées vx, vy et vz du vecteur « normale », en format libre, séparés par un espace et/ou une tabulation. Il n’est pas nécessaire, en entrée, que ce vecteur soit normalisé à 1. En sortie, par contre, OVNI normalise ce vecteur à 1.

Le mot clé <VECTEUR> (ou <VERT\_NORMAL> en version anglaise) permet d’entrer un tableau de normales aux sommets. Il n’y en a pas dans l’exemple ci-dessus. Il est suivi, sur la même ligne, du nombre de normales aux sommets. Ce nombre peut être égal au nombre de sommets, mais ce n’est pas une obligation. En effet, certains sommets sont anguleux et comportent donc plusieurs normales. D’autres sont partagés par plusieurs facettes dont la surface est continue. La normale représente alors la normale au plan tangent à cette surface. Une normale peut donc être utilisée/partagée par plusieurs facettes.

L’entrée du tableau de vecteur est analogue à celle des points ou sommets.

Il y a autant de lignes que de valeurs annoncées sur la ligne du mot clé.

Chaque ligne comporte d’abord un entier utilisé comme numéro de normale (ou de vecteur), suivi des 3 coordonnées vx, vy et vz. En entrée, ce vecteur peut ne pas être normalisé à 1. En sortie, OVNI forcera la normalisation à 1.

Le mot clé <LUMINANCE> (idem en version anglaise) permet de définir les normales à affecter à chaque sommet de facette. Il est suivi du nombre de facettes pour lesquelles on entre des normales aux sommets. Il est souvent égal au nombre de facettes, mais ce n’est pas une obligation. Par exemple, une facette qui est plane et présente un angle avec ces voisines n’est pas forcément entrée ici. On peut éventuellement en chacun de ses sommets lui affecter la même normale (et donc le même indice de vecteur). En principe, les 3 valeurs vx, vy et vz devraient être le mêmes que celles de la normale au barycentre.

Il doit y avoir autant de lignes que de valeurs annoncées avec le mot clé. La syntaxe est analogue à celle des facettes. Le premier entier correspond au numéro de facette à laquelle sont affectées les normales. Le second nombre, toujours un entier, correspond au nombre de sommets de la facette. En cas de désaccord avec l’entrée correspondante dans le tableau des facettes, OVNI le signale. Suivent ensuite les indices (des entiers également) des différentes normales à prendre dans le tableau des vecteurs, autant d’indices que de sommets.

Le mot clé <ASPECT\_FACE> (ou <POLY\_ATTR> en version anglaise) permet d’entrer un certain nombre d’attributs aux facettes. Ce mot clé est suivi sur la même ligne du nombre de facettes pour lesquelles on entre des attributs. Ce nombre peut être différent, mais inférieur ou égal au nombre de facettes de l’objet.

Il est suivi d’un certain nombre de groupe de lignes, autant que le nombre annoncé avec le mot clé. Chaque groupe de ligne commence par le numéro de la facette à laquelle on affecte les attributs. Les différents attributs sont repérés par des sous-mots clés de ASPECT\_FACE. Seule la première ligne correspondant à une facette contient son numéro. Les lignes suivantes qui concernent la même facette ne répètent pas ce numéro. Seul le sous-mot clé et la valeur correspondante changent.

Les sous-mots clé reconnus sont :

* <GROUPE> (ou <GROUP> en version anglaise) pour introduire la notion de groupe connue dans Crira.
* <CODMATFACE> (ou <POLY\_REF\_ATTR> en version anglaise) pour introduire un numéro de matériau.

Les autres sous-mots clés du format SDM sont ignorés.

Le mot clé <POSITION> (ou <PLACEMENT> en version anglaise) permet d’entrer une matrice de transformation de l’objet. C’est une matrice 4\*4 permettant de traiter des rotations, des déplacements latéraux, des symétries et également un facteur d’agrandissement. Le format SDM prévoit de pouvoir entrer plusieurs matrices. Le mot clé est suivi du nombre de matrices (1 dans l’exemple ci-dessus).

Les lignes suivantes contiennent ces matrices, repérées d’abord par un numéro, autant de numéros que de matrices. Seule la dernière entrée sera retenue par OVNI. L’usage de plusieurs matrices est une façon de dupliquer un objet, et sur chaque copie on y applique une matrice différente. Par exemple on peut n’avoir qu’un seul objet « Missile » dans la base de données, mais plusieurs missiles identiques sous les ailes de l’avion, chacun à une position différente. On peut aussi n’avoir qu’une aile de l’avion en base de données, la seconde étant obtenue par symétrisation. Actuellement OVNI ne supporte pas cette fonctionnalité de duplication, ce cas ne s’étant pas présenté.

Le numéro est suivi du sous-mot clé <MAT\_POSITION>, lui-même suivi des 4 premières valeurs de la matrice. Les 3 lignes suivantes ne comportent que les valeurs suivantes de la matrice, à raison de 4 valeurs par ligne.

D’autres possibilités sont prévues dans le format SDM, mais OVNI ne reconnaît pour le moment que le mot clé <MAT\_POSITION>. Par exemple, la matrice entrée dans l’exemple ci-dessus permet de transformer un cube en un parallélépipède rectangle en jouant sur le second terme de la seconde ligne qui vaut 0.25 au lieu de 1, puis un déplacement de 2 en Z. Mettre seulement des 1 sur la diagonale de la matrice et des 0 partout ailleurs revient au même que de ne pas mettre de matrice.

1. Fichiers Object File Format (.off)

C’est un format très simple, un des tous premiers mis au point en 1986 chez Digital Equipment Corporation. C’est un fichier de type texte, en ascii, qui ne comporte qu’un seul objet, un seul tableau de points et un seul tableau de facettes. Il n’y a pas de notions de normales, de groupes, de matériaux,…

Typiquement, un fichier .off représentant un cube est construit ainsi :

OFF bois.jpg

8 6 0

1.0 0.0 0.0

0.0 1.0 0.0

0.0 0.0 1.0

0.0 0.0 0.0

1.0 0.0 1.0

1.0 1.0 0.0

1.0 1.0 1.0

0.0 1.0 1.0

4 0 5 6 4

4 1 5 6 7

4 2 4 6 7

4 3 0 4 2

4 3 0 5 1

4 3 1 7 2

La première ligne contient l’identificateur OFF. En option, on peut avoir un nom de fichier image qui sera utilisé comme texture (mais non pris en compte par OVNI).

La ligne suivante contient 3 entiers : le nombre de sommets (ici 8), le nombre de facettes (ici 6) et un 3ème en réserve.

Viennent ensuite les coordonnées des sommets : ici 8 lignes avec les coordonnées x, y et z des points. Ce sont des réels, en format libre et séparés par un caractère blanc (espace). Le premier triplet correspond au premier point, le second triplet au second point et ainsi de suite.

Viennent ensuite les lignes contenant la description des facettes, avec en début le nombre de sommets de la facette (ici partout 4, mais ce nombre n’est pas imposé), puis les numéros d’indices des points de chaque sommet, autant que de nombre de sommets (la numérotation commence à 0). Ce sont des entiers en format libre et séparés par un espace.

OVNI lit ce type de fichier. Il peut aussi le créer. On le fera via le menu Fichier/Enregistrer sous… Il faudra choisir le type \*.\* dans le menu déroulant et entrer explicitement l’extension .off dans le nom de fichier. C’est volontairement qu’on ne propose pas d’office cette extension via le menu déroulant, car l’usage de ce type de fichier n’est qu’anecdotique. Néanmoins, le code existe dans OVNI. À déconseiller sauf pour un usage précis, comme l’utilisation d’un logiciel comme Monomoff ne connaissant que ce format en entrée et pour y tester un code de subdivision géométrique de facettes. Mais comme toute l’information sur les groupes, les normales, … est perdue et l’intérêt de ce format est faible, du moins en enregistrement.

À noter qu’on trouve sur Internet, notamment sur le site de l’Inria, soit directement des fichiers de type .off, soit de façon indirecte, des fichiers ne contenant d’un tableau de points dans un premier fichier et qu’un tableau de facettes dans un second fichier, les deux étant interdépendants. Construire un fichier .off à partir des deux fichiers de points et de facettes est très simple en utilisant un éditeur de texte : il suffit de mettre bout à bout les deux fichiers en y ajoutant une ou deux lignes d’informations. Ce peut être une solution pour bâtir une base de données qui n’existe pas sous d’autres formats exploitables plus directement.

1. Fichiers Wavefront Obj (.obj)

C’est un format de type texte piloté par des lettres clés placées en première colonne. OVNI ne peut que lire ce type de fichier, pas l’enregistrer.

Il est très répandu dans les modèles 3D. C’est une modèle basé sur l’utilisation de points, de facettes, d’objets et connaissant les normales aux sommets.

À la différence des autres formats présentés ici, la notion de point est commune à tous les objets : il n’y a qu’une seule numérotation des points, valable pour tous les objets, même si le tableau de points est entré en plusieurs fois. L’avantage, c’est qu’un même point peut être utilisé par plusieurs objets, ce qui facilite les calculs de normales aux sommets communs à ces objets. L’inconvénient est qu’on perd l’indépendance totale des objets. Dans OVNI, l’indépendance des objets étant la règle, on s’arrange pour que dès la lecture du fichier, les points communs pour le fichier OBJ mais appartenant à plusieurs objets, soient en fait dupliqués. La méthode originale utilisée dans OVNI consistait, dans un premier temps, à dupliquer tout le tableau de points dans tous les objets, puis dans un second temps, objet par objet, à simplifier la BDD en éliminant dans chacun d’eux les points inutilisés. Cette méthode était simple à mettre en œuvre, mais a posé problème sur de grosses bases de données avec beaucoup de points et beaucoup d’objet, car on a besoin temporairement de beaucoup d’espace mémoire. Dans quelques cas particuliers, cela conduisait même à un dépassement de la capacité d’adressage de Windows 32 bits. C’est la raison qui a motivé l’introduction d’une méthode de lecture optimisée de ces fichiers, proposée en option dans le menu Fichier/Préférences. C’est maintenant la méthode par défaut, mais l’ancienne méthode est toujours présente et activable car la nouvelle est optimisée en occupation de place mémoire, mais est plus longue à l’exécution.

Comme pour le format SDM d’Oktal-SE, OVNI ne supporte qu’un sous-ensemble des possibilités du format Wavefront OBJ. Les fonctions reconnues, la plupart ne nécessitant qu’une ou deux lettres suivies de paramètres séparés par des espaces, sont :

* v : suivi de coordonnées en x, y et z de points ;
* vn : suivi de coordonnées en vx, vy et vz d’une normale au sommet ;
* f : défini une facette. Cette lettre est suivie de plusieurs groupes d’entiers, autant de groupes que de points définissant une facette. Il n’y a pas de limitation sur le nombre de points par facette. S’il y a beaucoup de points dans une facette, les groupes d’entiers peuvent continuer sur la ligne suivante, mais sans répétition de la lettre f. À l’intérieur de chaque groupe, il peut y avoir de 1 à 3 valeurs entières, séparées par un /.  Le premier entier, le seul indispensable, donne le numéro de point du sommet (à chercher donc dans les entrées spécifiées par la lettre v). Le second entier, s’il est présent, donne le numéro de point à utiliser pour un début de texture (entré via le mot clé vn, mais non utilisé dans OVNI). Le troisième entier, s’il est présent, donne le numéro de la normale au sommet (à chercher donc dans les entrées spécifiées par les lettres vn). Si l’entier lu est positif, sa valeur indique un indice de tableau dans un des tableaux des points v, vt ou vn (la numérotation commence à 1). Si l’entier lu est négatif, il indique un indice relatif par rapport au dernier point lu avec l’option v correspondante. On peut par exemple obtenir des lignes comme ceci :
  + f 4 3 5 la facette utilise 3 points dont les indices sont dans cet ordre 4, 3 et 5 (lus par v). Il faut compter le nombre pour savoir combien de point il y a dans la facette ;
  + f 4/5/9 3/8/12 5/25/4 la facette utilise 3 points dont les sommets sont d’indices 4, 3 et 5 (lus par v). Les textures utilisent les points 5, 8 et 25 (lus par vt mais ignorés dans OVNI) et les normales les indices 9, 12 et 4 (lus par vn);
  + f 4//9 3//12 5//4 idem au cas précédent mais pas de texture
  + f -10 -11 -9 Indices relatifs par rapport au dernier numéro lu.
* fo : est équivalent à f ;
* usemtl : les facettes qui suivent ce mot clé utiliseront le nom du matériau décrit sur cette ligne
* g : suivi d’un nom donné à un groupe de facettes, interprété comme un objet par OVNI.

Il existe d’autres mots clés, non reconnus par OVNI. Citons par exemple vt pour entrer les coordonnées d’un point à utiliser comme point de départ dans un fichier de texture, un certain nombre de mots clés relatif à l’utilisation de surfaces courbes comme curv, curv2, surf, la possibilité d’utiliser des surfaces de Bézier, la possibilité d’avoir un fichier comportant le même objet sous plusieurs résolutions avec le mot clé lod,…

Il existe aussi un mot clé s pour « smoothing group » permettant de définir si des facettes sont planes, et dans ce cas il faudra ne pas tenir compte des normales aux sommets si elles ont été entrées, ou doivent être considérées comme courbes. Cette information n’est pas exploitée actuellement par OVNI.

Ce format ne connaît en entrée que les normales aux sommets. C’est à OVNI qu’il reviendra la tâche de calculer la normale au barycentre de chaque facette.

1. Fichiers Autodesk (.3ds)

C’est un format binaire. Il est lu indifféremment par toutes les machines, qu’elles soient de type big endian ou little endian, la conversion étant directement implémentée dans lib3ds, de façon transparente pour l’utilisateur.

On trouvera des informations détaillées sur ce format sur Internet, en particulier sur et via le site de téléchargement de la librairie lib3ds (voir le paragraphe sur l’installation et en particulier dans l’installation sous Linux / Sun Solaris §6.2.3).

Ce format est basé sur une notion de « mesh », qu’OVNI interprète comme des objets. Il peut y avoir plusieurs meshes regroupés au sein de « nodes ». En particulier, un même « mesh » peut être utilisé plusieurs fois à l’intérieur d’un même « node », la différence ne se situant qu’au niveau d’une matrice de transformation qui va permettre de créer un nouvel objet à partir du premier en y appliquant des déplacements, des symétries, des rotations et/ou un facteur d’échelle. Inutile donc, par exemple, d’introduire les coordonnées des points pour les 2 ailes d’un avion : une seule aile détaillée suffit dans le fichier, la seconde sera calculée par symétrie. Dans la pratique, on n’a rencontré ce type de construction que sur très peu de bases de données 3ds. La construction d’un autre objet par déplacement fonctionne dans OVNI 3.27, tous les objets sont bien créés mais la symétrisation ne fonctionne pas toujours (cf. fichier a6intruder.3ds).

Comme dans le format SDM, les objets sont indépendants, et chacun possède son tableau de points et son tableau de facettes. Dans ce format, les facettes ont obligatoirement 3 sommets, jamais plus. La récupération de ces tableaux est confiée à des fonctions de lib3ds. Les normales au barycentre sont récupérées via une fonction de lib3ds. Par contre, c’est OVNI qui calcule les normales aux sommets des facettes. Il semble exister dans lib3ds des fonctions qui peuvent permettre de calculer les normales aux sommets, mais jusqu’à présent, elles n’ont pas été utilisées avec succès.

Ce format connaît aussi la notion de « smoothing » entre les facettes qui permettrait d’activer facette par facette un lissage de Gouraud. Cette information n’est pas exploitée par OVNI.

Toutes les informations d’un fichier 3ds relatives à la couleur des facettes, la semi-transparence, au plaquage de texture, etc., sont ignorées par OVNI.

Très souvent, on constate dans OVNI, que des normales de facettes sont inversées, typiquement une sur deux. D’autres logiciels de lecture comme Deep Exploration n’ont pas ce problème ou le contournent automatiquement. Il y a peut-être une information non exploitée par OVNI qu’il faudrait décoder dans ces cas-là.

1. Fichiers Polygon Files de Niratam (.ply)

*A noter que bien que l’extension .ply soit la même, le format Geo-Niratam est différent de celui développé par Gregory Turk à l’université de Stanford. (Voir ci-dessous § A.7 - ).*

C’est un format de type texte utilisé par Niratam, Code OTAN de calcul du rayonnement infrarouge de cibles aériennes. L’extension .ply n’est ici que pour identifier le type de fichier car elle n’est pas imposée par Niratam. De toutes les façons, ce n’est pas sur l’extension qu’OVNI choisi le type de traitement à appliquer en lecture sur un fichier .ply, mais sur le contenu du fichier dont il cherche à reconnaître quelques caractéristiques.

À la différence d’autres formats, les coordonnées des points d’une facette sont données avec chacune des facettes. Ainsi, un même point, soit mêmes x, y et z est entré plusieurs fois dans le fichier. De ce point de vue, le format n’est pas optimal, par contre l’information est assez concentrée et chaque facette est totalement indépendante des autres. Ce format ne connaît que les points, les facettes et les objets. Indirectement, on trouve une notion de matériau qu’on peut assimiler aux groupes de Crira. Ce format ne possède aucune autre information, ni normales, ni couleurs, ni textures.

Dans beaucoup de fichiers, peut être en raison des divers changements de formats passant d’un monde Unix à un mode PC, les terminaisons de lignes ne sont pas homogènes. Il n’y a parfois que le caractère CR (Carriage Return), parfois que le caractère LF (Line Feed, standard Unix), parfois le couple de caractère CR/LF (standard Windows). Cela se traduit par la lecture de lignes vides, qu’il convient d’ignorer.

Un fichier de ce type comporte en première ligne les 3 lettres GEO. Le format reconnu par OVNI a été testé sur des fichiers de version v3.0.

La seconde ligne du fichier contient le nom de l’avion, du missile éventuellement suivi d’un commentaire (séparé par un caractère blanc).

La troisième ligne contient le mot clé ITTEM suivi du signe = et d’un entier.

La quatrième ligne contient le mot clé NEXH suivi du signe = et d’un entier donnant le nombre de tuyères (Number of Exhausts).

Les lignes suivantes contiennent les coordonnées en x, y et z des centres de tuyères. Il y a autant de lignes que de valeurs données par NEXH. En tête de chaque ligne, on trouve un mot clé comme COEX1, COEX2, … (1 pour 1ère tuyère, 2 pour 2ème tuyère,…) suivi du signe = et des coordonnées x, y et z.

Le mot clé NEXH et les lignes suivantes peuvent être absents. Dans ce cas, il n’y a pas de tuyères et on a donc affaire à autre chose qu’un avion ou un missile : un bâtiment par exemple.

Le début des entrées de données est indiqué par une ligne contenant le mot Start, généralement : \*\*\*\*\*\* Start of Geometry panel data \*\*\*\*\*\*

À partir de là, la structure du fichier est répétitive par groupe jusqu’à la fin.

Chaque groupe comporte au début le nombre de données du groupe.

L’interprétation des lignes suivantes dépend de la valeur lue sur la ligne ITTEM.

* Si cette valeur vaut 2, on lit ensuite un « Material Number ». C’est un entier dont la signification est spécifique à Niratam. Pour OVNI, cette valeur sera interprétée pour définir un numéro de groupe au sens Crira.
* Si cette valeur vaut 5, on lit ensuite sur la même ligne deux chaînes de caractères, dont la première de 3 lettres est SKN, ITO, ITI ou JCV, et la seconde représente un nom de matériau par exemple. La première des deux chaînes servent à coder un numéro de groupe au sens Crira, mais pas forcément au sens Niratam. La seconde est ignorée. Vu le peu de fichiers qu’on possède sous ce format, on n’a pas cherché de raffinements, mais cette seconde chaîne pourrait être exploitée si besoin.

Vient ensuite une ligne contenant tout d’abord un numéro de facette puis un nom d’objet auquel appartient cette facette. C’est à OVNI de faire le tri dans les noms d’objets car chaque facette étant indépendante, les facettes d’un objet donné ne sont pas forcément groupées ensemble.

Viennent ensuite plusieurs lignes contenant chacune les 3 coordonnées x, y et z des sommets de facettes. Le nombre de sommets à lire est déterminé à l’aide de la première valeur n lue dans chaque groupe de données. Il y en a (n-2)/3.

Ce sera ensuite à OVNI de calculer les différentes normales, cette notion n’étant pas connue dans ce format.

1. Fichiers Polygon Files de l’université de Stanford (.ply)

Documentation extraite de Wikipédia :

<https://fr.wikipedia.org/wiki/Format_de_fichier_de_polygones>. A noter que la version anglaise ne contient pas tout à fait les mêmes informations !

Le code intégré à OVNI est issu des travaux de Gregory Turk, téléchargés depuis : <https://www.cc.gatech.edu/projects/large_models/ply.html>

The [Le Digital Michelangelo Project](http://graphics.stanford.edu/projects/mich/) [[archive](http://archive.wikiwix.com/cache/?url=http%3A%2F%2Fgraphics.stanford.edu%2Fprojects%2Fmich%2F)] à l'[Université Stanford](https://fr.wikipedia.org/wiki/Universit%C3%A9_Stanford) utilisa le format PLY pour un scanner 3D de très haute résolution de "[David](https://fr.wikipedia.org/wiki/David_(Michel-Ange))", sculpture de [Michelangelo](https://fr.wikipedia.org/wiki/Michelangelo).

**PLY** est un [format de fichier](https://fr.wikipedia.org/wiki/Format_de_fichier) informatique connu sous le nom de ***Polygon File Format*** (« Format de Fichier de Polygones ») ou le ***Stanford Triangle Format***.

Le format fut principalement conçu pour stocker des données tri-dimensionnelles provenant de [scanners 3D](https://fr.wikipedia.org/wiki/Scanner_tridimensionnel). Il consiste en une description relativement simple d'un objet unique comme une liste de polygones nominalement plats. Une grande variété de propriétés peuvent être stockées, telles que couleur et transparence, normales aux surfaces, coordonnées de texture. Le format permet d'avoir des propriétés différentes pour le devant ou l'arrière d'un polygone

Il existe deux versions de ce format de fichier, un en [ASCII](https://fr.wikipedia.org/wiki/ASCII), l'autre en [binaire](https://fr.wikipedia.org/wiki/Binaire).

***Note :*** Le format binaire peut se présenter sous 2 formes : ordre des octets en Little Endian ou Big Endian. Les codes originaux de Greg Turk lisent bien ce paramètre (2ème ligne de l’entête) mais ne le traitent pas. Sous Windows, c’est le codage Little Endian qui est normalement utilisé. Dans Ovni, si on détecte un fichier binaire de type Big Endian, on inverse l’ordre des octets pour chaque type de donnée lue (entiers sur 16, 32 bits, flottants sur 32, 64 bits…). Sous Linux, ce ne sera sans doute pas la même chose et il faudra adapter cette inversion d’octets : Peut se faire via une compilation conditionnelle concernant la variable swap\_endian dans ply.cpp !

**Le format de fichier**

Au début de chaque fichier ply, il y a un [header](https://fr.wikipedia.org/wiki/Header) indiquant le contenu du fichier

header // Début du header (n’existe pas dans aucun des fichiers testés !)

ply // Type du fichier (Ovni teste sa présence en 1ère ligne)

format ascii 1.0 // Fichier codé en ASCII

comment VCGLIB generated

.

. // Corps du header

.

end\_header // Fin du header

Exemples d'éléments que peut contenir ce header:

element vertex 100 // Cette ligne signifie qu'il y a 100 points

property float x // l'abscisse du point sur l'axe des x

property float y // l'ordonnée du point sur l'axe des y

property float z // la cote sur l'axe des z

property float nx // la normale a x

property float ny

property float nz

property uchar red // le rouge du code RGB

property uchar green // le vert

property uchar blue // le bleu

property uchar alpha // la transparence

element face 1000 // Cette ligne signifie qu'il y a 1000 faces

property list uchar int vertex\_indices // liste des nombres de points qui vont se trouver dans chaque facette

**Voir aussi**

* [MeshLab](https://fr.wikipedia.org/wiki/MeshLab) : Application open source Windows, Mac OS X et Linux, pour visualiser, traiter et convertir des [maillages](https://fr.wikipedia.org/wiki/Maillage_(structure_de_donn%C3%A9es)) tri-dimensionnels depuis ou vers le format PLY.
* [CloudCompare](https://fr.wikipedia.org/wiki/CloudCompare) : Application open source Windows et Linux, pour visualiser, traiter et convertir des [maillages](https://fr.wikipedia.org/wiki/Maillage_(structure_de_donn%C3%A9es)) ou nuages de points tri-dimensionnels depuis ou vers le format PLY.
* [Mathematica](https://fr.wikipedia.org/wiki/Mathematica) Une méthode de calcul fonctionnant avec les fichiers PLY.

**Références/Liens**

* Greg Turk. ["The PLY Polygon File Format"](https://web.archive.org/web/20161204152348/http:/www.dcs.ed.ac.uk/teaching/cs4/www/graphics/Web/ply.html).
* [PLY - Polygon File Format](http://paulbourke.net/dataformats/ply/)
* [Quelques outils pour travailler avec les PLY](http://www.cc.gatech.edu/projects/large_models/ply.html)
* [Une bibliothèque pour lire et écrire les PLY](https://web.archive.org/web/20081203195143/http:/www.cs.princeton.edu/~diego/professional/rply/) ([Autre lien](http://w3.impa.br/~diego/software/rply/))
* [Another C++ software library for reading and writing PLY files (GPL 3.0 license)](http://www-sop.inria.fr/members/Thijs.Van-Lankveld/prog/ply/doc/index.html)
* [Une banque de modèles 3D au format PLY](http://graphics.stanford.edu/data/3Dscanrep/)

1. Fichiers Milkshape 3D (.m3d)

C’est un fichier de type texte. Le logiciel Milkshape 3D est un shareware pouvant lire en entrée divers formats de données, dont les fichiers .obj et .3ds. Il peut en sortie exporter ce qu’il a lu dans un fichier texte. L’extension n’est pas imposée, mais par commodité nous avons choisi .m3d plutôt que .txt afin de différentier ces fichiers. Dans un premiers temps, ces fichiers n’étaient pas lus directement par OVNI. C’est pourquoi la lecture de ces fichiers issus de l’exportation de Milkshape 3D a été introduite dans OVNI. Aujourd’hui elle n’a plus beaucoup d’intérêt, mais elle est restée dans OVNI comme une possibilité supplémentaire.

Un fichier m3d possède une entête contenant la chaîne « MilkShape 3D ASCII » en première ligne.

Il connaît la notion d’objets, de facettes, de normales aux sommets des facettes et de matériaux. Les matériaux peuvent être affectés d’une couleur, mais cette information n’est pas exploitée dans OVNI. Comme dans le format 3ds, il n’y a que 3 points par facette.

Typiquement, la partie utile commence par la lecture du mot clé Meshes : suivie du nombre d’objets que contient la base de données.

Pour chaque objet, on trouve en première ligne son nom.

En seconde ligne, on trouve le nombre de points de l’objet suivie d’autant de lignes que de nombre de points. Dans chacune de ces lignes on y trouve entre autres données, les coordonnées x, y et z des points. Seules ces trois valeurs sont exploitées dans OVNI.

Ensuite, on trouve une ligne contenant un nombre de normales aux sommets des facettes puis les coordonnées x, y et z de ces normales sur autant de lignes que de normales. Il peut ne pas y avoir de normales. Dans ce cas le nombre de normales est nul.

Pour terminer, on trouve un nombre de facette suivi d’autant de lignes qu’il y a de facettes. Chaque ligne ne comporte que 8 entiers : la première valeur est ignorée, les 3 suivantes correspondent aux indices des 3 sommets dans le tableau des points et les 3 suivantes aux indices des normales dans le tableau correspondant pour chacun des sommets (à condition que le nombre de normales ne soit pas nul, sinon on y trouve des 0). Le dernier entier n’est pas exploité.

Les informations sur les matériaux pourraient être exploitées et transformées en numéros de groupe au sens Crira. Ce n’est pas le cas actuellement, le besoin n’étant plus d’actualité en raison du support direct des formats .obj et .3ds.

Construire manuellement un fichier .m3d est assez simple si on ne possède que des fichiers de points et de facettes interdépendants et peut se faire via un éditeur de texte. C’est une solution qui a été utilisée quelques fois. Maintenant que le format .off est supporté par OVNI, la construction y est encore plus facile. C’est ensuite OVNI qui génèrera un fichier .bdd mieux adapté à l’usage dans Crira.

1. Fichiers du Groupe 3D (.g3d)

Ce format est de type texte et plus particulièrement basé sur XML. Une description approfondie du format peut être trouvée dans les documents issus du groupe de travail G3D, en particulier [DR5] et [DR6]. Comme pour les autres formats, l’implémentation de G3D dans OVNI est partielle et ne concerne que les fonctions utiles à un transfert vers le format SDM d’Oktal-SE. Crira supportera très probablement dans le futur le format G3D en plus de SDM, mais ce n’est pas le cas aujourd’hui. D’autre part, cette implémentation n’a été testée que sur un nombre très restreint de bases de données d’avions, trois ou quatre. On peut donc s’attendre à découvrir des manques lors de la lecture d’autres bases de données. En particulier, le format G3D, tout comme le format OBJ par exemple et également le format SDM, supporte plusieurs niveaux de détails (ou Lod) pour un même objet. Cette fonctionnalité n’est pas reconnue dans OVNI. Au mieux, si de tels fichiers sont ouverts dans OVNI, on se retrouvera en fait avec plusieurs objets différents, de résolution différente, alors qu’en fait il s’agit des mêmes. L’introduction de cette fonctionnalité se fera dès que le besoin s’en fera sentir et qu’elle pourra être testée sur un fichier validé.

La version actuelle d’OVNI supporte les fichiers en version 2.0 du format g3d. La version 1 n’est plus supportée. Il y a des incompatibilités avec la version 2, mais surtout, vu le peu de fichiers concernés et qui de toutes façons ont été adaptés, un travail de compatibilité n’a pas d’intérêt dans ce cas précis.

D’un point de vue général, le format G3D XML s’apparente au format SDM, mais il a en plus une notion de balise de fin pour chaque balise entrée, par exemple <objet id=1….> marque le début de la définition de l’objet numéro 1 et </objet> la fin de cette définition.

Le décodage, l’analyse et le contrôle de validité sont confiés à Expat au travers d’une librairie libexpat. C’est un parseur (ou analyseur) XML, disponible sur Internet (voir le paragraphe sur l’installation d’OVNI et plus particulièrement §6.2.4).

Les mots clés reconnus et le décodage associé dépendent de leur profondeur dans l’arborescence du format XML (valeur XML\_Depth pour Expat). On peut trouver effectivement le même mot clé à plusieurs profondeurs différentes (par exemple : normales\_s). C’est la profondeur dans l’arborescence qui différentie le traitement qui en est fait.

Nous ne détaillerons ici que les grandes lignes du décodage et seulement les mots clés connus par OVNI dans la version actuelle.

Le mot clé <types\_valeurs> (XML\_Depth=1) va permettre de récupérer par la suite des numéros de groupe au sens Crira ou des types de matériaux au sens de la donnée codmatface du format SDM.

Le mot clé <objets> (XML\_Depth=1) sert à connaître le nombre d’objets dans la base de données.

Pour chacun d’entre eux, il existe un mot clé <objet> (XML\_Depth=2) permettant de récupérer un identificateur ou un numéro d’objet, ainsi qu’un nom d’objet.

Au même niveau, on récupère la définition de <type\_materiau> et <type\_valeur> qui seront utilisés pour coder respectivement les différents matériaux et les différents numéros de groupes au sens Crira.

Pour chaque objet on peut récupérer

* Via une profondeur XML\_Depth=5
  + Le nombre de sommets via le mot clé <sommets> ;
  + Le nombre de facettes via le mot clé <facettes> ;
  + Le nombre de normales aux sommets <normales\_s> ;
* Via une profondeur XML\_Depth=6 ;
  + les coordonnées x, y et z de sommets avec le numéro d’indice correspondant via le mot clé <sommet> ;
  + les coordonnées vx, vy et vz de normales aux sommets avec le numéro d’indice correspondant via le mot clé <normale\_s> ;
  + un numéro de facette via le mot clé <facette>
* Via une profondeur XML\_Depth=7 et pour chaque facette ;
  + les indices des sommets de chaque facette via le mot clé <sommets> ;
  + les 3 coordonnées de la normale au barycentre de chaque facette via le mot clé <normale\_b> ;
  + les indices des normales aux sommets de chaque facette via le mot clé <normales\_s> ;
  + un type de matériau via le mot clé <materiau> ;
  + un numéro de groupe Crira via le mot clé <valeurs> ;

On peut noter que les informations de profondeur XML\_Depth=3 et XML\_Depth=4 ne sont pas exploitées pour le moment. C’est ce qui concerne notamment le nombre de maillages de l’objet (ou le nombre de niveaux de détails) ainsi que le point d’entrée de ces différents maillages (ou lod) à proprement parler.

1. Fichier .STL

Description issue du site https://en.wikipedia.org/wiki/STL\_(file\_format)#searchInput

STL est le format propriétaire du logiciel “**[st](https://en.wikipedia.org/wiki/Stereolithography" \o "Stereolithography)**[ereo](https://en.wikipedia.org/wiki/Stereolithography" \o "Stereolithography)**[l](https://en.wikipedia.org/wiki/Stereolithography" \o "Stereolithography)**[ithography](https://en.wikipedia.org/wiki/Stereolithography" \o "Stereolithography) [CAD](https://en.wikipedia.org/wiki/Computer-aided_design) software” créé par « [3D Systems](https://en.wikipedia.org/wiki/3D_Systems) ». On le nomme aussi "**S**tandard **T**riangle **L**anguage" ou "**S**tandard [**T**essellation](https://en.wikipedia.org/wiki/Tessellation) **L**anguage". Ce format de fichier est supporté par plusieurs autres logiciels 3D. Il est largement utilisé pour des prototypages, l’impression [3D et la conception assistée par ordinateur](https://en.wikipedia.org/wiki/3D_printing). Les fichiers STL ne décrivent que la géométrie de la surface d’un objet tridimensionnel sans représentation de couleur, texture ou d’autres attributs communs des modèles de CAD. Le format STL se décline en 2 représentations : [ASCII](https://en.wikipedia.org/wiki/ASCII) et [binaire](https://en.wikipedia.org/wiki/Binary_file). Les fichiers binaires sont plus répandus car plus compacts.

Un fichier STL décrit une surface via des facettes triangulaires à l’aide d’une normale unitaire et de sommets des triangles utilisant un système de coordonnées cartésiennes. La spécification originale demandait à ce que toutes les coordonnées soient positives, mais cette restriction a été levée et des coordonnées négatives sont couramment rencontrées dans les fichiers STL actuels. Les fichiers STL ne contiennent pas d’informations d’échelle, et les unités sont arbitraires.

**ASCII STL**

Un fichier STL en ASCII commence par la ligne :

solid *name*

où *name* est une chaîne de caractères optionnelle (toutefois si *name* est omis, il doit tout de même y avoir un espace après solid). Le fichier se poursuit par un nombre quelconque de triangles, représentés ainsi :

facet normal *ni nj nk*

outer loop

vertex *v*1*x* *v*1*y* *v*1*z*

vertex *v*2*x* *v*2*y* *v*2*z*

vertex *v*3*x* *v*3*y* *v*3*z*

endloop

endfacet

avec chaque *n* ou *v* est un nombre flottant en notation, par exemple : "2.648000e-002".

Le fichier se termine par

endsolid *name*

**STL Binaire**

Les fichiers STL ASCII peuvent devenir très gros, c’est pourquoi une version binaire de STL existe. Un fichier binaire STL commence par un entête de 80 caractères (qui est généralement ignorée, mais ne devrait normalement pas commencer par "solid" car cela peut amener certains logiciels à les considérer comme des fichiers STL ASCII). À la suite de l’entête, on trouve un entier non signé sur 4 octets (little endian) donnant le nombre de facettes triangulaires contenues dans le fichier. Puis suivent les données décrivant à leur tour chaque triangle. Le fichier se termine simplement après le dernier triangle.

Chaque triangle est décrit par 12 nombres flottants codés sur 32-bits : 3 pour la normale puis trois autres pour chacune des coordonnées X/Y/Z de chaque sommet – tout comme dans la version STL ASCII. Ensuite vient un entier non signé sur 2 octets ("short") dénommé "attribute byte count" dans le format standard : ce devrait être zéro car la plupart des logiciels ne l’interprète pas (c’est le cas actuellement pour Ovni).

Les nombres flottants sont représentés avec la convention IEEE floating-point et sont supposés être en « little endian », bien que ce ne soit pas dit dans la documentation.

UINT8[80] – Header

UINT32 – Number of triangles

foreach triangle

REAL32[3] – Normal vector

REAL32[3] – Vertex 1

REAL32[3] – Vertex 2

REAL32[3] – Vertex 3

UINT16 – Attribute byte count

end

**Codage de la couleur en STL binaire**

Il existe au moins deux extensions officieuses du format STL binaire permettant de coder la couleur des facettes :

* Les logiciels VisCAM et SolidView utilisent les deux octets à la fin de chaque triangle pour coder l'information RVB sur 15 bits :
  + les bits 0 à 4 codent l'intensité du bleu (0 à 31),
  + les bits 5 à 9 codent l'intensité du vert (0 à 31),
  + les bits 10 à 14 codent l'intensité du rouge (0 à 31),
  + le bit 15 est à 1 si la couleur est définie, ou à 0 si la couleur n'est pas définie.
* le logiciel Materialise Magics utilise l’entête de 80 octets pour représenter la couleur de l'objet complet. Si une couleur est définie, alors l'entête définit la chaîne ASCII "COLOR=" suivie de 4 octets définissant l'intensité du rouge, vert, bleu et la transparence, dans un intervalle de 0 à 255. Cela définit la couleur de l'objet complet, mais chaque facette peut redéfinir sa propre couleur. Magics permet également de définir l'aspect d'un matériau et la réflexion de la surface. À la suite de la chaîne "COLOR="<rgba>, il peut y avoir la chaîne "MATERIAL=" suivie de trois couleurs (3×4 octets) : la première couleur correspond à la teinte de la réflexion diffuse, la deuxième à la réflexion spéculaire, et la troisième à la lumière ambiante. Ce codage du matériau est préféré au codage de la couleur de l'objet.

Le codage de la couleur d'une facette est donné par 2 octets :

* les bits 0 à 4 codent l'intensité du rouge (0 à 31) ;
* les bits 5 à 9 codent l'intensité du vert (0 à 31) ;
* les bits 10 à 14 codent l'intensité du bleu (0 à 31) ;
* le bit 15 est à 1 si la couleur est définie, ou à 0 si la couleur n'est pas définie.

L'ordre de définition rouge, vert et bleu est inversé dans les codages de ces deux familles de logiciels ce qui pose des problèmes de compatibilité des fichiers car un logiciel lisant ces deux fichiers ne peut les distinguer. Il n'est pas non plus possible de définir une facette avec une transparence propre.

***Note 1*** : Ovni ne décode l’entête qu’il considère comme un nom d’objet. Il tente de décoder les couleurs contenues dans le champ « Attribute » des fichiers binaires si des valeurs de couleur différentes de 0 existent : Ovni les lit, en crée une liste sans doublons et code un numéro de groupe et de matériau d’après leur rang trouvé dans la liste dès la lecture. Les 3 composantes de couleurs trouvées servent à modifier les tableaux internes de palette de couleurs, dans la mesure où leur nombre n’excède pas la capacité de ces tableaux, soit 32 couleurs. La vraie couleur lue est accessible via le menu ou le bouton de colorisation des matériaux. Si on décisre récupérer la palette de couleur standard, il suffit de recharger le fichier Default.pal.

***Note 2*** : le format STL, Ascii ou binaire, intègre directement dans chaque facette les coordonnées des 3 sommets. A la lecture, OVNI se contente de les lire et les numéroter. Ainsi, un même sommet (mêmes 3 coordonnées) se retrouve plusieurs fois dans la base de données interne (au moins 3 fois, ce qui contribue à produire une grosse base de données en interne). Cela suffit pour voir si la base est intéressante ou pas. Au cas où l’utilisateur décide de la conserver ou l’utiliser, il convient dans un premier temps de simplifier cette base (en jouant éventuellement sur la tolérance d’égalité des points), ce qui aura pour effet d’éliminer et regrouper les doublons de sommets. Il faudra ensuite recalculer toutes les normales pour obtenir des normales aux sommets, indispensables pour le lissage de Gouraud (ou de Phong). Il faudra aussi vérifier, via le menu « Sens des normales » que toutes les facettes sont bien parcourues dans le même sens (CCW en standard, colorisées en bleu violacé), changer leur sens de parcours, si besoin, puis recalculer les normales. L’enregistrement au format SDM sera alors plus optimal.

1. Tableau comparatif des contenus des fichiers 3D

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | SDM | G3D | OBJ | 3DS | M3D | PLY | PLY  Stanford | OFF | STL |
| Points/  Sommets en x,y,z | Oui | Oui | Oui | Oui | Oui | Oui | Oui | Oui | Oui |
| Facettes | Oui | Oui | Oui | Oui | Oui | Oui | Oui | Oui | Oui |
| Nombre de points par facette | - | - | - | 3 | 3 | - | - | - | 3 |
| Normales au barycentre des facettes | Oui\* | Oui\* | Non | Non | Non | Non | Non | Non | Oui |
| Normales aux sommets | Oui\* | Oui\* | Oui\* | Non | Oui\* | Non | Oui\* | Non | Non |
| N° de groupe | Oui\* | Oui\* | Non | Non | Non | Non | Non | Non | Non |
| N° de matériau | Oui\* | Oui\* | matériau | matériau | Non | autre | Non | matériau | Non |
| Plusieurs objets | Oui | Oui | Oui² | Oui |  | Non | Non | Non | Non |

Oui\* => non obligatoire

Nombre de points par facettes - => quelconque

Oui² => plusieurs objets possibles, mais un seul tableau commun des sommets

Matériau / autre : une indication de nom ou autre (température) interprétée comme un numéro de matériau puis de groupe.

Normales au barycentre : Pour les formats qui n’en n’ont pas, on les recalcule à partir des sommets de facettes. Si plus de trois sommets, pas forcément dans un même plan, le calcul donne une normale moyenne. Pour le format 3ds, c’est une fonction interne de lib3ds qui fournit cette normale.

1. Calcul des normales dans OVNI
2. Calcul des normales au barycentre des facettes

Le calcul des normales au barycentre des facettes peut être fait par OVNI au cas où ces normales ne sont pas fournies dans le fichier de base de données. Ce peut être parce que le format de fichier ne connaît pas cette notion (.obj, .ply, .m3d, .stl notamment), mais ce peut être aussi parce que la donnée est absente du fichier. Il est systématiquement utilisé lorsqu’on utilise l’option de recalcul des normales ou qu’on modifie certains points ou certaines facettes d’un objet.

Pour Crira, cette normale est utilisée pour calculer une direction de réflexion d’un rayon venant de l’observateur et se réfléchissant sur la facette (principe du parcours inverse de la lumière). On saura ainsi quel type de rayonnement se réfléchit spéculairement sur cette facette, en particulier s’il provient d’un fond de ciel, d’une direction proche de l’horizon, d’un fond terrestre. C’est le cas lorsqu’on considère la facette comme plane dans Crira. Dans le mode plus évolué de lancer de rayons, permettant de traiter les réflexions multiples et les BRDF, on passera plutôt par un calcul de normales interpolées utilisant les normales aux sommets des facettes (voir ci-dessous).

Actuellement, le calcul de la normale au barycentre est fait à partir de 3 points, les 3 premiers en général. Si une facette est dégénérée et n’a que 2 points (arrive dans quelques bases de données), le calcul n’est pas fait et un message d’avertissement est généré. Au cas où les trois points retenus sont alignés (ou quasi alignés), OVNI choisi d’autres points dans la mesure du possible. Si aucun triplet ne convient, un message d’avertissement est généré. Dans ce cas, il conviendra de regarder de plus près cette facette, voir si une permutation circulaire des sommets arrange le problème. Cette méthode n’est valable que sur des bases de données qui ont des facettes à plus de 3 sommets. Au cas où aucune solution ne puisse être trouvée, c’est probablement que la facette est très particulière, par exemple très allongée. Il conviendra alors de la subdiviser ou même de l’éliminer et s’arranger pour boucher le trou produit, via les diverses options d’OVNI permettant de créer de nouveaux points et facettes. À noter que depuis la version 6.0 de Crira, il faut que les facettes soient triangulaires. C’est une contrainte imposée par le moteur de rendu interne dans Crira. Le calcul de normale à la facette est alors trivial. L’intérêt d’une solution de calcul permettant d’obtenir une normale moyenne, en cas de facettes non planes, calculée à partir de plus de trois sommets n’est plus d’actualité.

1. Calcul des normales aux sommets des facettes

Ce calcul, dans OVNI, nécessite la connaissance des normales aux barycentres des facettes qui utilisent un même sommet du même objet. Il sera effectué si la donnée n’est pas fournie dans le fichier, soit parce que format de fichier ne connaît pas cette notion, soit parce que la donnée est absente du fichier. Il est systématiquement utilisé lorsqu’on utilise l’option de recalcul des normales ou qu’on modifie certains points ou certaines facettes d’un objet.

Un des problèmes qu’on rencontre avec ce calcul automatique, est que les objets sont indépendants. Ainsi, un même point x, y, z, utilisé par deux objets différents, sera considéré en fait comme distinct. On ne pourra pas, en l’état actuel d’OVNI, utiliser les propriétés de facettes pourtant adjacentes utilisant ce sommet.

Les normales aux sommets sont utilisées, dans OVNI, pour obtenir un lissage de Gouraud (ou Ombrage de Gouraud, traduction de Gouraud Shading). Dans Crira, elles sont utilisées pour approximer les normales en chaque pixel d’une image, par un algorithme équivalent à celui de Phong (voir A.1 - Généralités). Dans le rendu en 3D, l’application du lissage de Gouraud donne un aspect plus continu aux diverses surfaces.

|  |  |
| --- | --- |
| image  Image facettée | image  Image lissée |

Le lissage de Gouraud conserve les bords de l’objet. Ainsi, l’aspect anguleux du bord de la sphère modélisée ci-dessus par 10 méridiens et 10 parallèles subsiste. Par contre, la surface interne lissée à droite est visuellement assez proche de ce qu’on obtiendrait en générant une sphère non lissée mais avec beaucoup plus de méridiens et de parallèles, leurs nombres étant choisis pour qu’une facette occupe de l’ordre de quelques pixels à l’écran. Dans ce cas, la base de donnée devient énorme et plus difficile à gérer, sauf à adopter un format de fichier autre, et gérant directement les surfaces : les facettes planes sont une représentation commode des surfaces lisses complexes, mais avec des limitations.

L’image ci-dessous montre quelques-unes des facettes de la sphère sur laquelle sont tracées les arêtes des facettes, les normales aux barycentres (à gauche et en bleu) et les normales aux sommets (à droite et en vert).

|  |  |
| --- | --- |
| image  Normales aux barycentres | image  Normales aux sommets |

Une des façons les plus simples de calculer une normale au sommet est de passer par une moyenne des normales de toutes les facettes utilisant ce sommet, en la normalisant le résultat à 1 en fin de calcul. Cette façon de faire fonctionne très bien si toutes les facettes sont à peu près de même taille. Par contre les résultats sont nettement moins bons en cas de maillage hétérogène conduisant à ce qu’un même sommet soit partagé par des facettes grandes et petites. Deux méthodes de pondération sont alors couramment utilisées : la première utilise l’angle au sommet des facettes, la seconde les surfaces des facettes. Elles donnent visuellement des résultats assez semblables. C’est la seconde méthode qui est utilisée dans OVNI, notamment parce qu’elle demande un peu moins de calculs : il faut un arc sinus en plus pour extraire la valeur de l’angle à partir d’un produit vectoriel.

Ce calcul est effectué par OVNI systématiquement pour tous les sommets, soit directement à la lecture du fichier si les normales aux sommets n’y sont pas, soit à la demande de l’utilisateur, lors d’un recalcul des normales.

Ce calcul simple présente toutefois un défaut. Il existe des parties d’objets pour lesquelles la facette doit être considérée comme plane ou qu’il vaut mieux la considérer comme telle car l’application de l’algorithme conduit à des défauts de visualisation si deux facettes doivent être considérées comme anguleuses entre elles. Par exemple, si on génère un cube dans OVNI, les facettes de part et d’autre des arêtes su cube ne doivent pas être lissées entre elles, sinon on obtient un rendu s’apparentant à celui d’une sphère, ce qui, dans l’ensemble, n’est pas souhaitable.

Afin d’identifier automatiquement ce type de facettes dans OVNI, on introduit une notion de seuil angulaire. C’est aussi ce type de solution qui est utilisée dans des logiciels comme Deep Exploration.

Au moment d’affecter au sommet d’une facette la normale au sommet calculée, on la compare avec celle au barycentre de la facette :

* Si l’angle entre les deux normales est inférieur au seuil fixé, on conserve la normale au sommet calculée ;
* Si l’angle est supérieur au seuil, la facette doit être considérée comme anguleuse, et en ce sommet, on affectera la normale au barycentre de la facette.

Typiquement, la valeur de l’angle de seuil est de 35°. C’est une valeur qui est utilisée par défaut dans OVNI, tout comme dans Deep Exploration. On entre cette valeur dans une fenêtre obtenue via le menu Ficher/Préférences. Elle est valable pour toute la base de données. Elle est modifiable à la volée, l’application sur la visualisation étant faite en temps réel (du moins sur des bases de données pas trop grosses et avec un ordinateur suffisamment puissant).

Dans l’ensemble, cette méthode de seuillage donne de bons résultats. Ainsi, les bords de fuite des ailes sont maintenant corrects car il n’y a pas de raison de lisser ensemble les bords supérieurs et inférieurs des ailes.

|  |  |
| --- | --- |
| image  Image facettée | |
| image  Lissage sans seuillage : problème en bord de fuite | image  Lissage avec seuillage de 35° : correct |

Toutefois, il existe des cas où le résultat n’est encore pas celui escompté. Ainsi, un bord de tuyère (image ci-après) reste anguleux alors qu’on le souhaiterait lissé, au moins pour le groupe des facettes externes de la tuyère (celles du dessus ici) et pour le groupe des facettes internes (en dessous, non vues ici). Par contre, les deux groupes dessus-dessous présentent un angle entre eux. Notons que cet effet est présent parce que les facettes externes et internes de la tuyère appartiennent au même objet. Si elles étaient dans deux objets différents, on n’aurait pas ce problème. Néanmoins, trop subdiviser en sous-objets, n’est pas non plus une bonne solution car leur indépendance, si elle est favorable comme dans ce cas, peut l’être nettement moins dans d’autres, là où on voudrait que les surfaces présentent un aspect continu.

|  |
| --- |
| image |

Le calcul de la normale au sommet va se faire, dans OVNI, par étapes successives. Pour chaque sommet, OVNI établi dès la lecture du fichier, une liste des facettes qui utilisent ce sommet. Ce sont des facettes adjacentes, appartenant au même objet. Comme on l’a déjà vu, OVNI ne sait pas traiter, en l’état actuel, des points communs à deux objets (ou plus). Pour une facette donnée et un sommet donné, on initialise le calcul de la normale au sommet avec la normale au barycentre de la facette en cours de traitement. Ensuite, on compare chaque normale au barycentre des facettes adjacentes avec la normale au barycentre de la facette sur laquelle on est en train de travailler. Si l’écart angulaire, entre la normale au barycentre de la facette en cours et celle de la facette adjacente, est supérieur à un seuil, on ne tient pas compte la normale à la facette adjacente. S’il est inférieur, on conserve cette normale. En fin de calcul, on moyenne les normales retenues en y appliquant un coefficient de pondération lié à la surface de chaque facette. La méthode est donc différente de ce qui a été vu précédemment car on ne tient pas compte systématiquement de toutes les normales des facettes qui utilisent un sommet. Elle est cependant complémentaire, la contrainte pouvant être moins restrictive. L’angle de ce second seuil est entré comme une valeur multiplicative du premier seuil. Dans le cas de la tuyère évoqué ci-dessus, il suffit d’un facteur de 0.5 pour voir l’effet de lissage. D’un point de vue plus général, c’est plutôt une valeur proche de 2 qu’il faudrait entrer (tests sur un cube par exemple). Dans la pratique, après avoir testé sur une dizaine de bases de données, un bon compromis semble être une valeur de 1,6. C’est la valeur entrée par défaut. Cette valeur est cependant ajustable et sauvegardée dans le fichier Application1.ini.

Comme précédemment, et pour peu que l’ordinateur soit suffisamment puissant, on peut modifier ce facteur à la volée dans la fenêtre d’entrée et observer en temps quasi réel l’effet sur la base de donnée. Il ne faut toutefois pas donner une valeur trop forte sinon on observe d’autres effets indésirables, qui sont les mêmes, ou très proches de ceux vus sans seuillage et donc en ignorant que des facettes puissent être anguleuses entre elles (voir figure ci-dessous).

|  |  |
| --- | --- |
| image  Second seuil : 1,6 (soit 35\*1.6=56°) | image  Second seuil : 2,6 (soit 35\*2.6=91° > 90°) |

Les exemples ci-dessous montrent l’influence du seuil angulaire sur le sommet d’un cône, ce qui s’apparente au nez d’un avion.

|  |  |
| --- | --- |
| image  Image facettée | |
| image  Lissage sans seuils | image  Lissage sans seuils, normales affichées |

L’image montrant le lissage de Gouraud sans seuil est très belle mais fausse. En effet, la partie inférieure du sommet du cône est de la même couleur que la partie supérieure alors que la source de lumière est en haut à droite, ce que montre bien l’image facettée. La normale au sommet est dans l’axe du cône ici (petit trait en bleu sur la droite).

Si maintenant on applique le seuil de 35° on obtient les images ci-dessous. Le sommet du cône possède maintenant des normales multiples. Notons que l’aspect au voisinage du sommet reste facetté. Dans ce cas, c’est normal car les facettes sont triangulaires et au sommet, il n’y a pas de continuité possible des normales entre une facette et sa voisine.

|  |  |
| --- | --- |
| image  Lissage avec seuil de 35° | image  Seuil de 35°, normales affichées |

On peut alors se dire qu’un cône peut se générer à partir d’un cylindre dont on réduit à un point, par changement d’échelle, un des deux cercles générateurs. Dans ce cas, le sommet des triangles qui est confondu avec le sommet du cône, possède bien maintenant deux normales, et donc la continuité des normales entre deux facettes adjacentes est assurée. Néanmoins, visuellement, cela ne change rien dans OVNI, l’aspect est le même que ci-dessus à gauche : peut-être un problème dans OpenGL qui n’arrive pas à bien gérer ce cas un peu particulier d’un sommet de triangle à 2 normales, ce qui pour l’algorithme de Gouraud revient à avoir un seul pixel mais 2 couleurs différentes pour ce pixel ; on part effectivement d’un cylindre bien homogène, mais plus on diminue la surface de ce qui deviendra la pointe, plus celle-ci se facettise comme le montre la série d’images ci-dessous.

|  |  |
| --- | --- |
| image  Cylindre généré à partir de 16 points / cercle | image  Diminution d’un cercle générateur : échelle 0.5 |
| image  Diminution du cercle générateur : échelle 0.25 | image  Diminution du cercle générateur : échelle 0.0625 |

Toutes les modifications ci-dessus peuvent se faire en temps quasi réel dans OVNI. Pour cela, OVNI repart systématiquement des normales aux sommets calculées comme indiqué précédemment. Quand le résultat est visuellement satisfaisant, il faut alors enregistrer ces normales modifiées dans la base de données afin qu’elles puissent être utilisées plus tard par Crira. C’est la raison de la présence, dans la fenêtre accessible via Menu/Préférences, de la case à cocher intitulée « Enregistrement dans la Bdd des normales aux sommets seuillées », non cochée par défaut, afin de pouvoir tester la meilleure configuration pour chaque base de données.

1. Création d’un nouveau panneau sous wxSmith

On décrit ci-dessous les étapes pour créer, via wxSmith dans Code::Blocks, un nouveau panneau accessible via un menu. En l’occurrence, il s’agit du menu « Zoom Spécifique » qui est pris comme exemple. Le style des étapes à suivre est volontairement télégraphique.

Menu C::B wxSmith, Ajouter wxDialog

Remplir le nom de la classe : par exemple ZoomSpecifique

Fichier d’entête et source seront créés dans le sous répertoire src avec le même nom

Accepter l’intégration dans les sous-projets (debug, release, …)

Fichier .wxs affiché en mode wxSmith

Lui donner un titre puis …

Entrer les différents éléments graphiques, zones de texte, cases à cocher, …

Il est préférable de les entrer dans l’ordre, du haut vers le bas, pour faciliter la lecture directe du fichier wxs (en fait un format xml), mais cela n’a pas d’importance.

Créer dans wxFrame, Outils le sous menu Zoom Spécifique, donner un nom à la variable, par exemple ici : Menu\_ZoomSpecifique (on peut garder le nom donné par défaut, mais ce sera moins parlant).

Ajouter dans { } un EVT\_MENU, ajouter un gestionnaire et accepter la proposition de nom (ou la changer). ***Note*** : bug C::B car il faut le faire 2 fois pour que ça marche (mais la seconde fois le nom change un peu, alors refuser le nom et choisir le bon ) !

Dans OvniMain.cpp est créé :

void OvniFrame::OnMenu\_ZoomSpecifiqueSelected(wxCommandEvent& event)

Ne pas hésiter à enregistrer les wxs modifiés et/ou .cpp, .h car, en tous cas pour les wxs, l’enregistrement ne semble pas toujours fait automatiquement.

Il faut remplir cette routine par au moins un show() afin d’afficher la boite de dialogue.

Ici, par exemple on va mettre : ZoomSpecifique\_Panel->Show();

Dans OvniMain.h, ajouter en haut un #include ”ZoomSpecifique.h” et ajouter un peu plus bas :

ZoomSpecifique\* ZoomSpecifique\_Panel;

Avec les autres déclarations du même type.

Dans ZoomSpecifique.h, ajouter #include ”OvniMain.h” après les headers introduits par wxSmith et avant la déclaration de classe.

En fin de ZoomSpecifique.h, ajouter OvniFrame\* MAIN; juste avant DECLARE\_EVENT\_TABLE, dans la zone private.

Dans ZoomSpecifique.cpp, ajouter à la fin de ZoomSpecifique::ZoomSpecifique :

this->MAIN = dynamic\_cast<OvniFrame\*>(parent);

Dans OvniMain.cpp créer la classe ZoomSpecifique\_Panel dans OvniFrame::OvniFrame, là où il y a déjà les autres déclarations :

ZoomSpecifique\_Panel = new ZoomSpecifique(this,wxID\_ANY);

Dans interface.h, ajouter en début (avec les autres déclarations du même type) :

class ZoomSpecifique;

Créer dans ZoomSpecifique.wxs un évênement OnClose dans { }. Ajouter dans la routine ainsi créée un Hide() ;

Dans ZoomSpecifique.wxs, associer par exemple un des boutons (OK, Quitter, …). Dans le code créé ajouter :

// Boutton OK <=> OnClose

wxCloseEvent close\_event;

OnClose(close\_event);

Ainsi, un clic sur ce bouton aura le même effet qu’un clic sur la croix de fermeture, en haut à droite.

***Attention :*** il faudra convertir au moins ZoomSpecifique.cpp en ***utf8***, via Editer / Encodage du fichier / UTF-8, surtout si certaines entrées texte ont des accents. Par prudence, ce sont tous les fichiers .cpp qu’il faudra convertir en utf-8 pour éviter les soucis liés aux caractères accentués dans les chaines de caractère. (pas de problèmes dans les commentaires).

À ce niveau, une regénération complète doit fonctionner et un clic sur le nouveau menu (Zoom Spécifique) doit provoquer l’affichage de la boîte de dialogue. Les boutons Quitter (ou OK) et la croix de fermeture en haut, à droite doivent aussi fonctionner.

Reste à remplir les autres fonctionnalités.

Il peut être utile de créer dans interface (.cpp et .h) un pointeur vers le panel créé afin de faciliter des initialisations par exemple.

Dans interface.h :

ZoomSpecifique \*MZoomSpec= nullptr;

Dans interface.cpp

if (MZoomSpec != nullptr) {

MZoomSpec->SpinCtrl\_LSI->SetValue(lround(m\_gldata.rotx));

MZoomSpec->SpinCtrl\_LAZ->SetValue(lround(m\_gldata.rotz));

}

Dans OvniMain.cpp (OuvrirFichier) :

Element->MZoomSpec= ZoomSpecifique\_Panel;

Précautions : dans Code::Blocks, le fait de lancer une compilation, une génération, suffit pour que les fichiers cpp, h soient automatiquement enregistrés au préalable. Ce n’est pas le cas des fichiers .wxs pour wxSmith. Il est donc utile (voir nécessaire) de forcer leur enregistrement via, par exemple, un clic droit sur l’onglet de ce fichier.

1. Compilation de wxWidgets

La génération des wxWidgets est assez sensible au compilateur et aux options utilisées. Il est préférable d’utiliser le même compilateur et les mêmes options pour tout logiciel utilisant wxWidgets.

Depuis mi-2019, Code::Blocks utilise un support 2D activé par une option spécifique lors de la génération des wxWidgets. Cette option n’est pas directement accessible en ligne de commande. Elle doit être activée en éditant le fichier setup.h et en modifiant la ligne :

#define wxUSE\_GRAPHICS\_DIRECT2D 0

Par

#define wxUSE\_GRAPHICS\_DIRECT2D 1 // Modif pour CB 11701 et +

Ce fichier setup.h existe à 2 endroits dans la génération des wxWidgets pour Windows. C’est celui dans C:\wxWidgets-3.2.0\lib\gcc\_dll\mswu\wx (sous-répertoire mswu pour Windows - unicode) qu’il est important de modifier. En fait, lors de la première génération des wxWidgets, il est recopié de celui par défaut situé dans les sources de base dans C:\wxWidgets-3.2.0\include\wx\msw (msw pour Windows. Il se peut que ce setup.h n’existe pas encore ici : utiliser alors setup0.h, version basique. Seule la modification du 1er setup.h est indispensable, mais par précaution on peut aussi modifier ceux de base.

Cette modification ne sert pas dans Ovni, mais autant la faire de façon à n’avoir qu’une seule librairie wxWidgets, utilisable indifféremment pour générer Ovni et/ou Code::Blocks.

Exemple de fichier .bat pour la génération de wxWidgets 3.2.0, en 64 bits à placer dans C:\wxWidgets-3.2.0\build\msw :

REM Mettre MinGW64\bin avant MinGW32\bin

**set** PATH**=**C:\MinGW64\bin;**%PATH%**

REM Ci-dessous, rename (si besoin !) pour éviter une mauvaise interprétation de détection de la présence de sh.exe

**ren** C:\Utilitaires\_Msys-Unix\bin\sh.exe shx.exe

REM les strip ne marchent que si VENDOR=cb\_64, sinon les adapter

REM Nettoyage des compilations précédentes :

mingw32-make -f makefile.gcc USE\_XRC**=**1 SHARED**=**1 MONOLITHIC**=**1 BUILD**=**release UNICODE**=**1 USE\_OPENGL**=**1 VENDOR**=**cb\_64 clean

REM Génération

mingw32-make -f makefile.gcc USE\_XRC**=**1 SHARED**=**1 MONOLITHIC**=**1 BUILD**=**release UNICODE**=**1 USE\_OPENGL**=**1 VENDOR**=**cb\_64 CPPFLAGS**=**"-m64" CXXFLAGS**=**"‑fpermissive ‑fno‑keep‑inline‑dllexport ‑std**=**gnu**++**11 ‑Wno‑deprecated‑declarations" LDFLAGS**=**"-Wl,--allow-multiple-definition" LDFLAGS**=**"-m64"

REM Compactage des dll (nécessite l’utilitaire unix-like strip)>log.txt 2>&1

strip ..\..\lib\gcc\_dll\wxmsw32u\_gcc\_cb\_64.dll

strip ..\..\lib\gcc\_dll\wxmsw32u\_gl\_gcc\_cb\_64.dll

REM Rétablir le nom de sh.exe

**ren** C:\Utilitaires\_Msys-Unix\bin\shx.exe sh.exe

La compilation créée tout un ensemble de fichiers, ici dans C:\wxWidgets-3.1.2\lib\gcc\_dll par exemple :



À noter que les versions de wxWidgets évoluent. On peut donc avoir 314 (en 2020), 315 (prévue fin 2020), 32 (en juillet 2022), … suivant la version en cours d’utilisation.

1. Modification du code généré par wxSmith

Code::Blocks est livré avec un générateur de code d’interface graphique, wxSmith. Ce générateur est utilisé dans Ovni pour créer les différentes fenêtres de dialogues ainsi que la fenêtre principale contenant la visualisation 3D sous OpenGL. Jusqu’au début 2020, l’anticrénelage (antialiasing) des formes 3D n’était possible que via les options du driver de la carte graphique. Le seul anticrénelage disponible et géré par OpenGL dans Ovni était celui des lignes.

Depuis la version 3.0 de wxWidgets, on dispose de paramètres supplémentaires permettant, au moins à la compilation, de programmer 2 valeurs : WX\_GL\_SAMPLE\_BUFFERS et WX\_GL\_SAMPLES. La première vaut 0 ou 1, est en fait un booléen, et indique si on utilise le multi-échantillonnage sur 1 pixel pour obtenir l’anticrénelage. Par défaut, cette valeur, pour être compatible avec les versions précédentes de wxWidgets vaut 0 (non activé). La seconde valeur donne le nombre de pixels utilisés : en standard on peut mettre 1, 4 (2x2), 9 (3x3) ou 16 (4x4). Le code interne wxWidgets est assez tolérant car on peut y mettre d’autres valeurs comme 2, 3, … De lui-même, le code transformera en interne cette valeur en une valeur correcte (a priori, la valeur correcte inférieure à celle entrée). Le multi-échantillonnage consiste à découper chaque pixel en nxn sous pixels jointifs, calculer sur ce sous-pixel comment est vu le forme 3D (vu / non vu notamment) puis à faire la moyenne des résultats sur le pixel. Plus la valeur de n est élevée, meilleur est l’anticrénelage, mais plus long est le temps de calcul par pixel. La valeur de 4 est la valeur minimale pour obtenir un anticrénelage et s’avère suffisante dans la plupart des cas.

Le code géré par wxSmith ajoute ces paramètres et leur assigne des valeurs dès qu’on utilise wxWidgets en version 3 ou supérieure. Néanmoins sur certaines machines, en particulier celles possédant à la fois une puce graphique et une carte graphique évoluée, notamment des portables, cela peut conduire à des soucis à l’exécution si Ovni n’est pas déclaré comme devant utiliser la carte graphique évoluée. Lorsqu’Ovni utilise la puce graphique, il se peut que le support de ces nouveaux paramètres ne soit pas reconnu par le driver et fasse planter Ovni. Il est possible de revenir au fonctionnement d’avant en invalidant l’ajout des 2 nouveaux paramètres si besoin. L’idéal serait de modifier wxSmith pour cela, et donc Code::Blocks lui-même. Mais ce n’est pas forcément souhaitable car d’un usage plutôt rare. Dans Ovni, on a donc choisi une solution alternative :

* on laisse wxSmith générer son code automatiquement,
* on invalide la génération automatique de code localement,
* on modifie légèrement le code généré pour détecter le cas particulier de non support des paramètres.

Au cas où on doive modifier la fenêtre graphique bridée, comme ajout d’une option, d’un bouton, … il suffit de : revalider la génération automatique, laisser faire les modifications utiles par wxSmith puis de nouveau invalider la génération automatique et réintroduire le code supplémentaire.

Dans Ovni, tout se passe dans OvniMain.cpp et dans OvniFrame::OvniFrame. On a ajouté le code suivant en tout début :

**int** test\_GLCanvasAttributes\_1[]={ WX\_GL\_SAMPLE\_BUFFERS,1,WX\_GL\_SAMPLES,4,0,0 };

**bool** GLExtend = **false**;

**if** (wxGLCanvas::IsDisplaySupported(test\_GLCanvasAttributes\_1)) GLExtend=**true**;

Ces trois lignes permettent de tester si les nouveaux paramètres sont supportés par la carte graphique et/ou son driver. Elles peuvent rester dans le code indépendamment de la génération automatique de code par wxSmith.

Le code généré par wxSmith vient ensuite et est encadré par 2 lignes de commentaires spécifiques :

//(\*Initialize(OvniFrame)

… code généré

//\*)

Pour empêcher wxSmith de générer le code automatiquement, il suffit de changer la première ligne de commentaire :

//(\*Initialize(OvniFrame)

par

//\_(\*Initialize(OvniFrame)

Soit juste ajouter un \_ avant la parenthèse. Noter que la colorisation de texte se réactive dans cette zone de code, colorisation qui avait été invalidée pour bien montrer que cette partie de code ne devrait pas, en principe, être modifiée.

Ensuite, après la déclaration du tableau GLCanvasAttributes\_1, il suffit d’ajouter les lignes :

if (!GLExtend) {

printf("WX\_GL\_SAMPLE\_BUFFERS non reconnu par la carte graphique et/ou son driver !\n");

GLCanvasAttributes\_1[6] = GLCanvasAttributes\_1[7] = 0;

}

juste avant le #if wxCHECK\_VERSION(3,0,0)*.*

Ces modifications restent actives tant qu’on ne restitue pas la gestion automatique du code par wxSmith, ce qu’on obtient en supprimant le \_ du premier commentaire. Mais dans ce cas, le code ajouté à la main sera supprimé. Il faudra donc recommencer.

Cette invalidation de génération automatique de code ne touche que la fenêtre principale, pas les autres dialogues.