# 

**SOMMAIRE** [1](#_Toc503777976)

[Remerciements 4](#_Toc503777977)

[**1-** **Présentation de l’entreprise** 5](#_Toc503777978)

[**2-** **Le DOTA** 6](#_Toc503777979)

[Contexte, Cahier des charges, planning et équipe 7](#_Toc503777980)

[**1-** **Contexte** 7](#_Toc503777981)

[**2-** **Cahier des charges** 7](#_Toc503777982)

[**3-** **Planning et équipe** 8](#_Toc503777983)

[Mission 9](#_Toc503777984)

[**1-** **Méthodologie** 9](#_Toc503777985)

[**2-** **Définition des données** 10](#_Toc503777986)

[**3-** **Interface** 12](#_Toc503777987)

[**4-** **Les fonctions de base** 15](#_Toc503777988)

[a. Lecture 15](#_Toc503777989)

[b. Sauvegarde 15](#_Toc503777990)

[c. Affichage 15](#_Toc503777991)

[**5-** **Les fonctions d’affichages** 16](#_Toc503777992)

[a. Affichage des sommets 16](#_Toc503777993)

[b. Affichage des arrêtes 16](#_Toc503777994)

[c. Affichage de la boite englobante 17](#_Toc503777995)

[**6-** **Les fonctions avancés** 17](#_Toc503777996)

[a. Sens des normales 17](#_Toc503777997)

[b. Fonction de soudage 18](#_Toc503777998)

[c. Fonction de division d’arrête 18](#_Toc503777999)

[d. Fonction de sélection de facette 19](#_Toc503778000)

[Points délicats rencontrés 20](#_Toc503778001)

[**1.** **WxWidgets et OpenGL** 20](#_Toc503778002)

[a. Apprentissage de WxWidgets 20](#_Toc503778003)

[b. OpenGL et orientation 20](#_Toc503778004)

[**2.** **Format de donnée BDD** 20](#_Toc503778005)

[Test et résultats 21](#_Toc503778006)

[**1.** **Test** 21](#_Toc503778007)

[a. Cube 21](#_Toc503778008)

[b. OVNI 21](#_Toc503778009)

[c. M2000 22](#_Toc503778010)

[**2.** **Résultats** 22](#_Toc503778011)

[Conclusion 23](#_Toc503778012)

[ANNEXE 24](#_Toc503778013)

# Remerciements

Je tenais personnellement à remercier   
Mr. Durand Gérard   
pour m’avoir donné ce stage et  
permis d’avoir passé un stage qui m’aura   
beaucoup appris.  
  
Je remercierai aussi mes collègues du DOTA  
qui ont toujours été une source de motivation  
et de bonne humeur.   
  
Enfin je remercie,   
Mr. Gely Denis  
car sans lui je n’aurai  
jamais eu ce stage.   
  
Présentation de l’ONERA

## **Présentation de l’entreprise**

L’Office National d’Études et de Recherches Aéronautiques est un établissement public français fondé en 1946 et placé sous la tutelle du ministère de la Défense. Le but était à l’époque de doter la France d’une aviation moderne.

De 1946 à 1962, l’ONERA a acquis une compétence indiscutable et apporta un concours inestimable aux constructeurs français d’avions et d’hélicoptères. Elle sut diversifier ses activités de recherche pour être aujourd’hui un des premiers acteurs français dans les domaines de l’aéronautique, de la défense et de l’aérospatiale.

Depuis sa création, l’ONERA représente la France dans la plupart des coopérations scientifiques internationales, notamment avec les laboratoires de recherche américains (NASA, US Air-Force, etc.), japonais, chinois et surtout avec ses homologues européens, maintenant réunis au sein de l’EREA (Établissements de Recherche Européens de l’Aéronautique) pour une politique de recherche intégrée.

La vocation de l’ONERA est de soutenir par la recherche et l’innovation l’industrie aérospatiale nationale et européenne. On retrouve parmi les clients-partenaires de l’ONERA, les grandes sociétés du secteur comme Airbus, Dassault Aviation, Thalès et Safran. Cet office est en grande partie financé (par contrats) par les industriels.

L’ONERA regroupe 2018 salariés, dont près de 300 doctorants et post-doctorants répartis sur 8 centres qui couvrent un large domaine de recherche, avec un budget de 233 millions d’euros. Ces centres se trouvent à :

-Palaiseau (siège social dans l’Essonne)  
-Châtillon (Haut de Seine)  
-Meudon (Haut de Seine)  
-Lille (Nord)  
-Salon de Provence (Bouche du Rhône)  
-Modane Avrieux (Rhône-Alpes)  
-Le Fauga-Mauzac (Midi-Pyrénées)  
-Toulouse (Midi-Pyrénées)

Pour ma part j’ai travaillé dans le DOTA (Département Optique Théorique et Appliquée au centre de Palaiseau dans le département MPSO (L’unité Modélisation Physique de la Scène Optronique) sous la supervision de M. Gérard Durand.

## **Le DOTA**

Le DOTA est un des plus importants centres européens de recherche en optronique de défense avec un effectif de 125 ingénieurs de recherche, 26 doctorants et post-doctorants, 15 techniciens.

Sa mission est le développement et la maîtrise des chaînes optiques, depuis la source jusqu’au post-traitement, dans une gamme de longueurs d’onde allant de l’infrarouge à l’ultraviolet, au service de l’aéronautique, de l’espace et des systèmes militaires.

Son périmètre s’est élargi récemment à certaines applications relevant de la sécurité et de l’environnement et, pour une petite part, à des préoccupations civiles, comme le domaine biomédical.

Le DOTA a la maîtrise de l’ensemble de la chaine optique, depuis la source jusqu’aux traitements, tant pour les aspects expérimentaux que pour les aspects de modélisation. Il s’appuie fortement sur le monde de la recherche universitaire et institutionnelle.

Pour une large part de son activité, le DOTA agit en tant qu’expert étatique. Il entretient des relations régulières avec les grands industriels du secteur. De plus, il a tissé des liens avec un ensemble de PME à travers lesquelles il valorise ses travaux.

L’activité du DOTA est structurée en quatre spécialités au sein du domaine de base "Senseurs pour la détection, l’observation et la mesure" :

-La physique de la scène optronique, qui regroupe la modélisation en sens direct passif et en actif, l’inversion des mesures, l’acquisition de données d’entrée nécessaires aux modélisations, et les campagnes expérimentales de validation.

-La spécification, la conception et la réalisation de dispositifs radiométriques multi spectraux, qui repose sur de nouveaux concepts instrumentaux, et sur le développement dans le domaine des nanotechnologies.

-L’imagerie à haute résolution, qui regroupe la propagation optique dans les milieux turbulents, l’optique adaptative et l’instrumentation multipupilles.

-Les lasers et leurs applications, qui regroupent la réalisation et la caractérisation de nouvelles sources, la réalisation et l’exploitation de nouveaux lidars, la guerre optronique, l’imagerie active et les nouveaux travaux sur le laser femtoseconde.

# Contexte, Cahier des charges, planning et équipe

## **Contexte**

Le logiciel OVNI de l’ONERA permet la visualisation et la correction de base de données 3D de multiples formats différents. Il a aussi pour but de faire le pont entre les bases de données géométriques que diffusent les fournisseurs et les logiciels de modélisation de l’ONERA qui ne fonctionnent pas forcément sur les mêmes formats de bases de données.

Le logiciel est actuellement fonctionnel et correspond bien au cahier des charges qui date de 2004. Les langages de programmation y sont le C, OpenGL, Tcl. L’interface entre ce dernier et le C s’appuie sur un logiciel tiers qui n’est plus maintenu et cela posait des problèmes sur la pérennité du logiciel OVNI qui voyait ainsi petit à petit sa durée de vie se réduire.

La mission consistait à comprendre le fonctionnement de l’ancien projet et de le transcoder dans de nouveaux langages et une interface qui sont mis à jour régulièrement pour permettre la durabilité du projet.

Le projet qui m’a été confié se divise en deux parties que j’ai développées :  
- Transcoder l’ancien projet OVNI et le passer de C/Tcl/OpenGL à C++/WxWidget/OpenGL.  
- Améliorer les fonctions existantes pour de meilleures performances et de meilleurs résultats.

## **Cahier des charges**

L’ancien code développé à partir de 2004 par des élèves de l’IUT d’informatique d’ Aix-en-Provence puis repris à l’ONERA se décomposait en trois fonctionnalités ou applications  :  
  
- Application 1 Visualisation : affichage de cellule, mode de visualisation (modification du point de vue, zoom), sélection de facettes et de groupes de facettes, de points, affichage des propriétés, rajout d’éléments, mode d’affichage, impression d’écran.

- Application 2 Fonction de travail sur fichier: ajout de facettes, calcul de la normale de chaque facette, détermination de la boîte englobante et calcul du meilleur angle de vue, calcul d’une image de rendu sur échantillonnée, calcul d’une image de rendu au format capteur, calcul de la surface apparente de chaque facette.

- Application 3 Outil interactif de visualisation d’image : récupération de l’image et affichage de celle-ci (image de rendu sur échantillonnée, image de rendu au format capteur), fonctions générales (zoomer sur l’image, afficher du texte sur l’image, information sur l’image, multi fenêtrage), interface, graduation (échelle), icônes.

Le logiciel OVNI intègre essentiellement l’application 1 et une bonne partie des codes développés dans l’application 2. Les codes développés pour l’application 3 sont intégrés dans les logiciels de modélisation du DOTA et sont en dehors du champ de ce travail.

## **Planning et équipe**

Le planning fut divisé en quatre temps sur les treize semaines équivalentes à la durée de mon stage. Cela se fit de la manière suivante :

**-Semaines 1 & 2 : Découverte de logiciel OVNI**

Dans un premier temps, il me fut essentiel de m’adapter à l’existant en testant le logiciel OVNI et en voyant toutes les fonctionnalités disponibles que j’aurai à transcoder. Ce fut aussi le moment de fixer certaines lignes directrices et de définir les améliorations et les priorités du projet.

**-Semaines 3 à 10 : Développement du logiciel OVNI WxWidget version**  
 Dans un second temps, qui est le plus conséquent, c’est le développement de la nouvelle   
 version du logiciel OVNI qui dura 9 semaines qui fut mon activité principale. Mais nous y reviendrons plus tard.

**-Semaines 11 à 12 : Correction et finition de l’existant**

Dans un troisième temps, nous avons corrigé et fait en sorte de rendre une version propre du code et suffisamment documentée pour permettre à toutes personnes de comprendre plus facilement le code du logiciel.

**-Semaines 13 : Ecriture rapport de stage**

Enfin dans un dernier temps, j’ai écrit le rapport de stage qui devait être rendu avant la fin du projet. Ce temps supplémentaire devait me permettre aussi de corriger des derniers bugs si besoin.

L’équipe quant à elle est composée de moi uniquement et de mon supérieur M. Durand qui supervise le projet et avec qui je discute chaque semaine des itérations suivantes.

# Mission

## **Méthodologie**

Après l’analyse de l’existant je me suis rendu compte que transcoder tout le code posait quelques soucis car la majorité du code à refaire faisait partie de l’interface, ce que je devais justement modifier. Je suis parti de ce constat pour refaire l’essentiel du code en ne reprenant que les algorithmes de base.

Du coup nous passons d’une structure dite dynamique (avec des pointeurs), à une structure automatique (basée sur des tableaux variables). L’objectif est d’avoir en priorité la possibilité de lire, modifier, sauvegarder rapidement et simplement, en ayant un algorithme plus simple à comprendre pour un suivi plus simple et dans le but d’obtenir un suivi et des modifications simples à faire. Le projet sera développé sous Codeblocks qui est l’IDE (L’environnement de développement) choisie pour le projet.

L’interface est faite via WxSmith, interface graphique à WxWidgets, qui s’utilise facilement sous Codeblocks car y est intégrée. Mon superviseur de stage M. Durand avait déjà commencé à travailler sur une maquette de ce projet. Il faut aussi savoir que WxWidgets est en partie basé sur le GTK qui fut un langage de type évènementiel que j’ai appris en seconde année de ma prépa intégrée au sein de l’ECE Paris. De ce fait je suis parti sur une méthodologie de type évènementiel et en y intégrant du MVC séparant ainsi modèle, vue et contrôleur.

Pour la présentation de ce travail je vais prendre chaque fonction du logiciel en expliquant au passage les soucis que j’ai eu et si le résultat est bien celui espéré, s’il est meilleur ou s’il est inférieur à celui de la version originale.

## **Définition des données**

Pour le cœur de l’application et le stockage de données je me suis inspiré du format BDD (format propriétaire de la société Oktal-SE, ou format SDM) en créant des classes pour chaque type de données (Sommet, Face, Normal, Luminance, Vecteur). Voici une simple définition et description de chaque donnée.

Sommet :

Ces données sont composées de 4 valeurs, la première servant à donner le numéro du point. Les trois autres sont les trois coordonnées XYZ du sommet. Il faut savoir que chaque sommet peut avoir plusieurs vecteurs (cela sera développé dans les parties ultérieures).

Face :

Ces données sont composées de 3 à N valeurs, la première étant le numéro de la facette, la seconde est le nombre de points de cette facette et les valeurs 3 à N sont les numéros des sommets de la facette. Dans la pratique N ≥ 5, car il y a au moins 3 sommets par facette. Il faut savoir que chaque facette possède une normale au barycentre de la facette. Il faut aussi signaler que les numéros de normales (voir ci-dessous) et de facettes coïncident. C’est aussi le cas pour les luminances (voir plus loin) : autant de luminances que de facettes.

Normal :

Ces données sont composées de 4 valeurs, la première étant le numéro de la normale au barycentre d’une facette et les autres sont les trois coordonnées XYZ de cette normale.

Luminance :

Ces données sont composées de 3 à N valeurs, la première valeur étant le numéro de luminance le second étant le nombre de luminances et les valeurs 3 à N sont les numéros des Vecteurs au sommet pour chaque facette. Il faut savoir que chaque numéro de face et de luminance coïncide.   
  
  
Vecteur :

Ces données sont composées de 4 valeurs, la première étant le numéro de vecteur et les 3 autres correspondent aux coordonnées XYZ du vecteur. Il faut savoir que le numéro de vecteur et le numéro de sommet ne coïncident pas forcément  : un sommet peut avoir plusieurs vecteurs en fonction de l’écart entre les différentes normales des faces qui l’utilise.

Pour dire les choses simplement, les vecteurs sont aux luminances ce que les sommets sont aux faces.

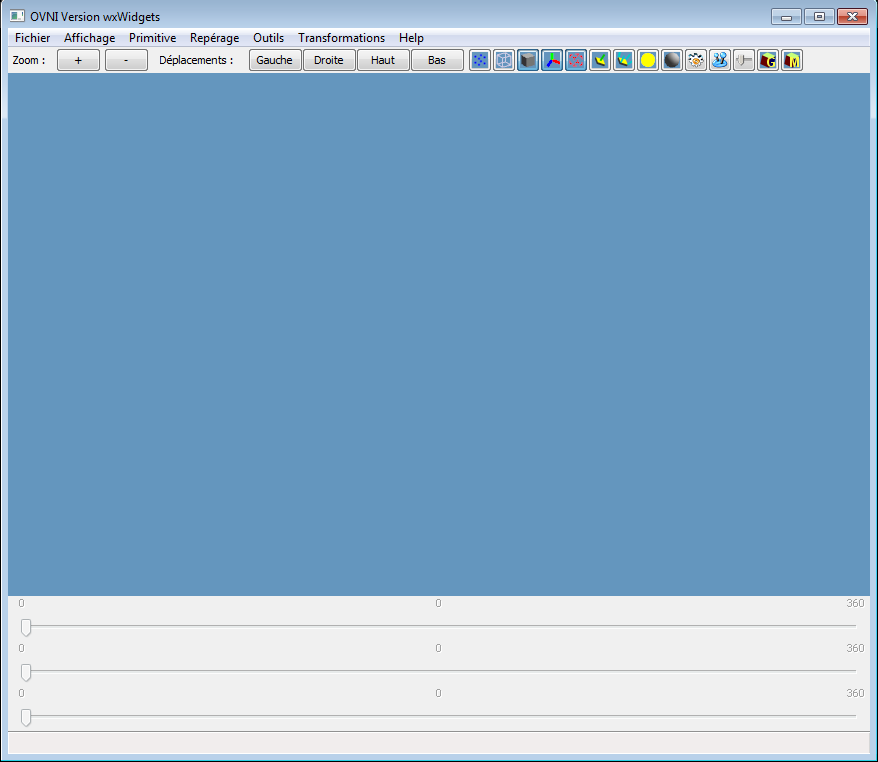
Une facette est considérée comme plane en générale et est composée à partir d’un certain nombre de sommets, au minimum 3. La normale à la facette est en fait la normale au barycentre de la facette. Si la facette est composée de 3 points, pas de soucis, la facette est plane et sa normale bien définie. Pour des facettes à plus de 3 sommets, on peut calculer une normale moyenne à partir d’une décomposition de la facette en triangles.

La notion de luminance est rattachée à l’algorithme de Gouraud, permettant de lisser la surface d’un objet en affectant à chaque sommet une normale, ici dénommée vecteur, ou normale au sommet. Les vecteurs sont calculés par une interpolation sur toutes les normales au barycentre des facettes utilisant un même sommet. Ces normales aux sommets sont utilisées par OpenGL et même le hardware des cartes graphiques pour donner un aspect lissé aux objets et ainsi simuler une surface continue.

Comme dit ci-dessus, un même sommet peut avoir plusieurs normales au sommet : tout dépend si ce sommet s’inscrit dans une continuité de surfaces (une seule normale) ou s’il doit y avoir une discontinuité anguleuse (plusieurs normales). Par exemple, sur un objet en forme de cube, les facettes latérales doivent être anguleuses entre elles. Le cube le plus simple possède 6 facettes et 8 sommets : il y aura donc 3 normales en chaque sommet de ce cube. Au contraire, sur une sphère approximée par tout un ensemble de facettes planes, l’ensemble de la surface doit paraître continu et il n’y aura qu’une seule normale en chaque sommet.

Dans les logiciels Onera, ces normales aux sommets vont être aussi utilisées pour calculer une normale en chaque pixel image, simuler une surface non plane (bien que décomposée en petites facettes planes) et ainsi mieux traiter les problèmes de réflexion de la lumière sur les surfaces (réflexion du soleil, réflexions multiples inter-surfaces, etc…). L’algorithme utilisé dans ce cas s’apparente à celui de Phong.

## **Interface**

L’interface est quasi identique avec celui de l’OVNI. Vous pouvez voir le design ci-dessous :

* : Affichage/Non affichage des points
* : Affichage/Non affichage des arrêtes
* : Affichage/Non affichage des faces
* : Affichage/Non affichage des axes X Y Z

: Affichage/Non affichage de la boite englobante

 : Affichage/Non affichage de la source lumineuse

 : Affichage/Non affichage des facettes avec la méthode de Gouraud

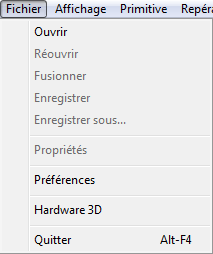
: Affichage/Non affichage de la fenêtre de modification



  : Affichage/Non affichage des sliders de modification XYZ en bas de la fenêtre

 : Affichage/Non affichage des facettes par groupe de couleur

 : Affichage/Non affichage des facettes par groupe de matériaux

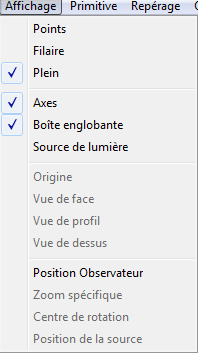


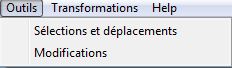
Ouvrir : Ouverture d’un fichier

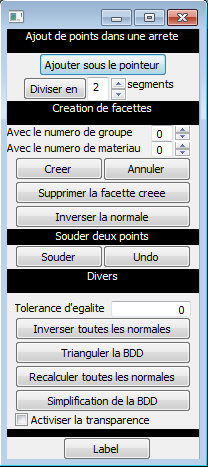
Préférences : Permet de paramétrer certains réglages mais n’a aucune conséquence dans le programme pour l’instant

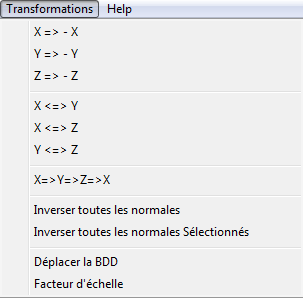
Hardware 3D : Affichage des paramètres de la carte graphique vus par OpenGL

Quitter : permet de quitter l’application

Points, Filaire, Plein, Axes, Source de lumière : Affichage/Non Affichage des points, arrêtes, faces, axes, boite englobante et source de lumière.

Sélections et déplacement : (non implémenté)  Affichage de la fenêtre sélection et déplacements qui permet la sélection et les manipulations d’objets de multiples façons.   
  
Modifications : Affichage de la fenêtre de Modifications ci-dessous.

Ajouter sous le pointeur : (non implémenté) Ajoute un point sous le pointeur de la souris dans une arrête.  
  
Diviser en : Permet de diviser une arrête en X segments de longueurs égales.  
  
Souder : Permet de souder deux points ensemble et affiche les points pour aider à la soudure.  
  
Undo : Annule la dernière modification de la base de données que ce soit souder/inverser les normales/etc…  
  
Inverser toutes les normales : la fonction inverse toutes les normales de la base de données   
  
Recalculer toutes les normales : cette fonction recalcule toutes les normales de la base de données.  
  
Simplification de la base de données : Cette fonction va simplifier la base de données, notamment si deux points sont au même endroit, elle recalcule toutes les normales/luminances/vecteurs et supprime les éventuels doublons

X => -X, Y => -Y, Z => -Z : ces fonctions inversent les signes des coordonnées de la base de données sur certains axes.  
  
X <=> Y, X< => Y, Y <=> Z, X => Y => Z => X : ces fonctions permutent certains des axes de la base de données.  
  
Inverse toutes les normales (sélectionnées) : Ces fonctions permettent d’inverser soit toutes les normales de la base soit seulement quelques-unes, celles qui sont en rouge, car elles ont été préalablement sélectionnées.

## **Les fonctions de base**

### Lecture

Pour la fonction de lecture j’ai utilisé des fonctions standard de WxWidgets, cela avait pour but de tester la capacité des fonctions de lecture de WxWidgets. Toutefois sur de longs fichiers ces fonctions posaient des difficultés sur certains des cas donnés par Mr. Durand. Pour la suite du projet il faudra faire en sorte de réécrire cette fonction de lecture pour utiliser plutôt les fonctions standard de C++ (en particulier CIN). Il faudrait aussi rajouter, comme dans OVNI d’origine, la fonctionnalité qui permettrait de travailler sur d’autres formats de fichiers de bases de données 3D car pour l’instant seule la lecture du format SDM (ou .bdd) a été implémentée, et ce toujours dans l’esprit de travailler sur un algorithme automatique. Il faudra impérativement penser à travailler les autres formats de bases de données pour les transposer au format SDM (.bdd) car c’est une des raisons d’être d’OVNI. La fonction LoadBDD occupe cette fonction pour les fichiers BDD.

Cette fonction d’ailleurs corrige la base de données s’il manque des éléments. Cette fonction fut d’ailleurs une plus des difficiles que j’ai eu à faire.

### Sauvegarde

La fonction de sauvegarde permet pour l’instant uniquement d’écrire des bases de données au format BDD. Elle réécrit toute la base de données du fichier sélectionné. Du point de vue rapidité elle est au moins aussi rapide que celle du logiciel OVNI précédent. La fonction SaveBDD occupe cette fonction.

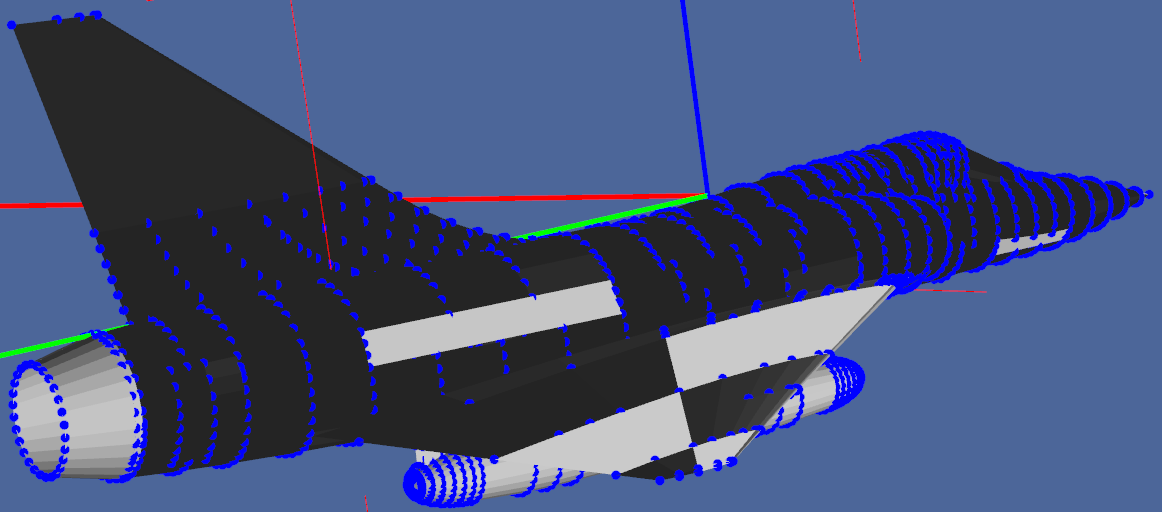
### Affichage

La fonction d’affichage appelée drawopengl est une fonction qui utilise les données de la base de données pour afficher avec une interface OpenGL avec l’aide des WxWidgets. Pour éviter de recharger en permanence la base de données dans OpenGL, on la charge au tout début ou seulement lorsque la base de données est modifiée. Des options supplémentaires existent, comme l’affichage des points, lignes, etc. Ces options sont elles aussi présentes dans la fonction drawaopengl avec des validateurs TRUE/FALSE pour rajouter ou pas ces éléments dans l’affichage en cours. Ces options seront développées plus loin.

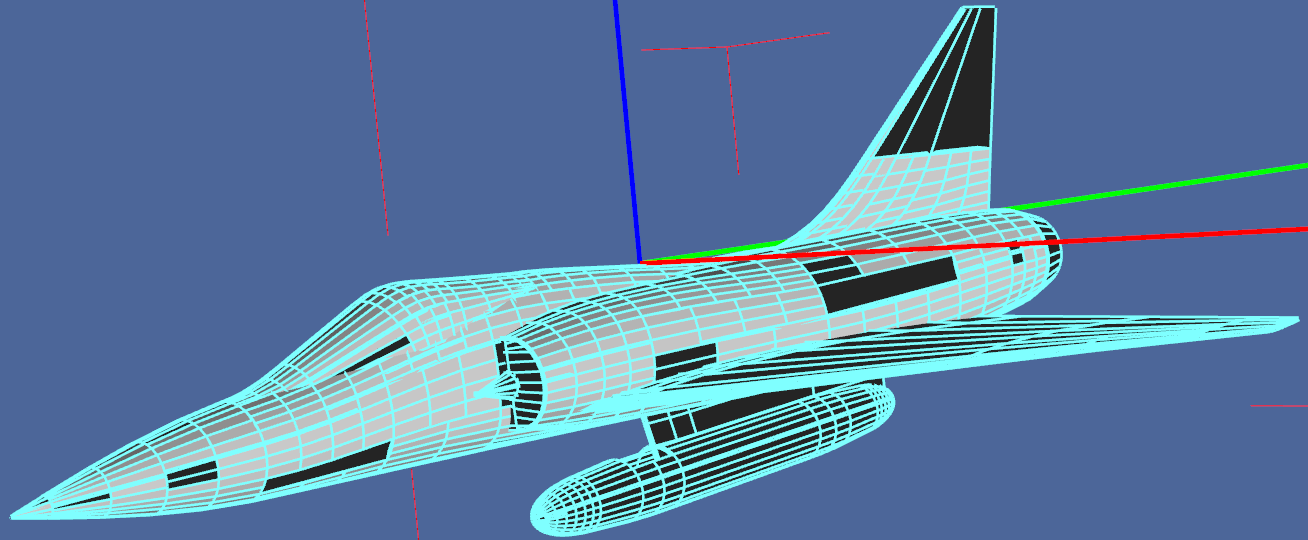
## **Les fonctions d’affichages**

## Affichage des sommets

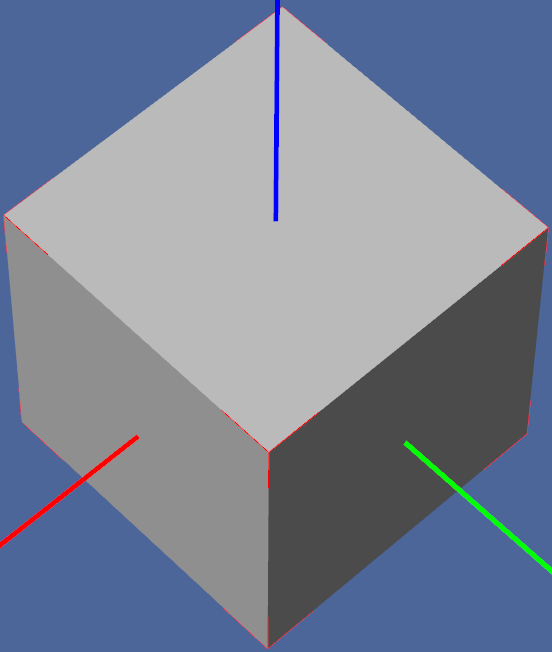
L’affichage des points, permet d’afficher chaque sommet



### Affichage des arrêtes



### Affichage de la boite englobante

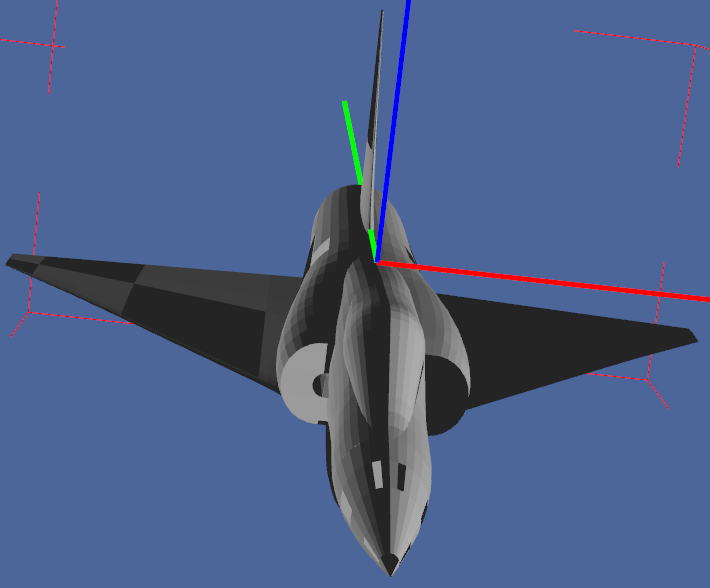
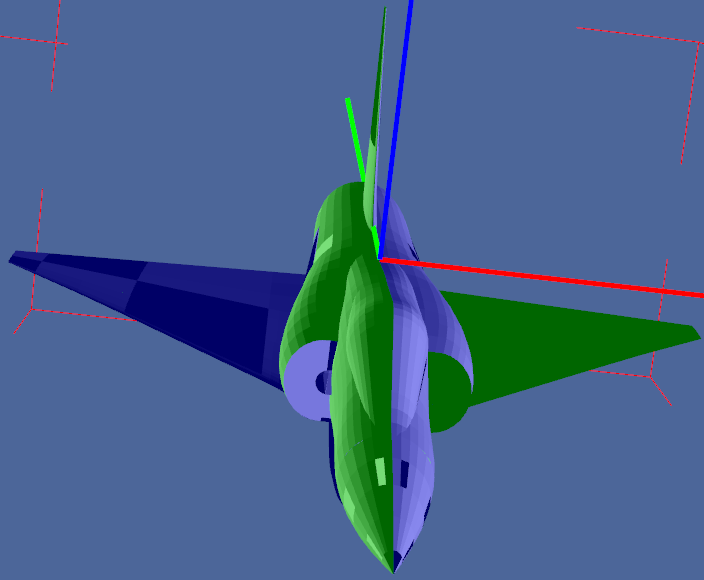
La fonction de boite englobante consiste à prendre les mesures maximales pour l’ensemble de la/les bases des données présentes à l’écran. Cette fonction est situé dans la fonction drawopengl qui affiche à l’écran la figure cela permet d’éviter de refaire une boucle et perdre du temps de calcul. Cette boite englobante est représentée par les petits traits en rouge (visibles ici sur les sommets du cube).

## **Les fonctions avancés**

### Sens des normales

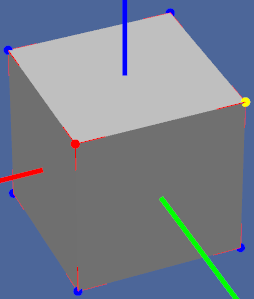
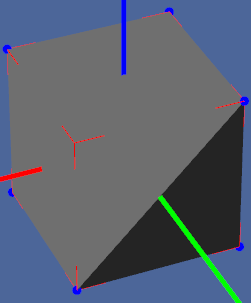
Cette fonction permet d’afficher le sens de parcours des points des facettes (corrélé au sens des normales) via deux couleurs :

-Bleu pour les facettes parcourues dans le sens horaire (selon OpenGL).  
-Vert pour les facettes dans le sens antihoraires.

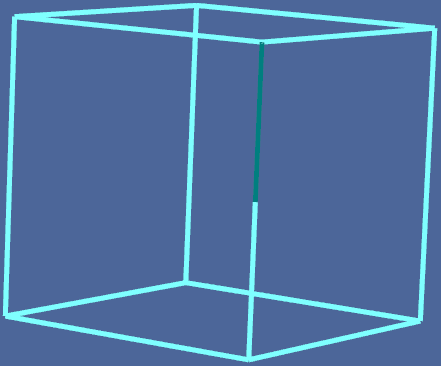
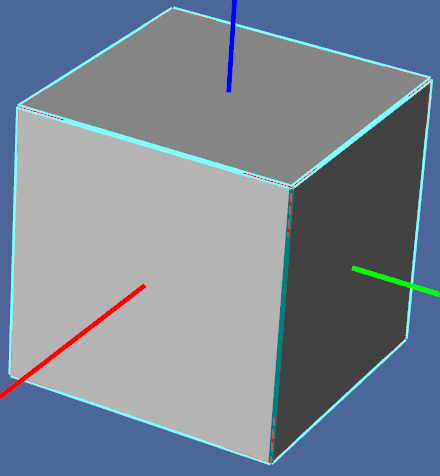
Cette fonctionnalité est aussi bien développée sur la version originale. Elle permet de voir les facettes qui ont été mal orientées et de les corriger avec la fonction inverser les normales.

## Fonction de soudage

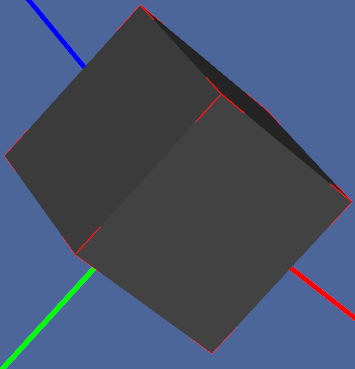
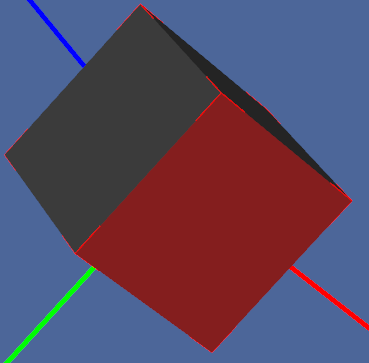
Cette fonction permet de souder deux points. Elle se fait en deux temps : elle affiche tout d’abord tous les points puis lorsqu’on positionne le curseur sur un premier point ce dernier devient jaune. Une fois sélectionné, il devient rouge. Ensuite lorsque le second point est sélectionné toutes les références au premier point sont transférées au second : comme on peut voir dans le cas ci-dessous.



## Fonction de division d’arrête

Comme pour la fonction de soudage cette fonction agit en plusieurs temps : dans un premier temps elle affiche les arrêtes comme sur les images ci-dessous puis après qu’on ait cliqué dessus, elle divise l’arrête en N morceaux de même longueur, dans le cas présent 2. Comme vous pouvez le voir sur l’image de droite.

## Fonction de sélection de facette

Cette fonction a pour but de sélectionner les facettes en leur donnant une couleur rouge bien reconnaissable parmi les autres couleurs utilisées. Les tons de luminances sont également respectés. Le résultat des deux versions d’OVNI sont identiques à ce propos même si la version d’OVNI classique réagit légèrement plus rapidement, la version WxWidgets permet, elle, de mieux sélectionner les facettes s’il y a un groupe à sélectionner.

# Points délicats rencontrés

## **WxWidgets et OpenGL**

### Apprentissage de WxWidgets

WxWidget est un langage peu documenté, ce qui posa quelques soucis surtout du point de vue de la correction des bugs. Cela me fit perdre de multiples heures. Mon expérience et les similitudes avec le GTK m’ont permis de réduire ces difficultés et les problèmes au minimum. Aussi l’interface WxSmith fut simple d’utilisation et en partie faite par Mr. Durand au préalable ce qui me fit gagner beaucoup de temps.

## OpenGL et orientation

Contrairement aux WxWidgets, je n’avais jamais eu d’équivalent avec l’OpenGL et ce fut la première fois que je codais pour faire du 3D. Ce fut à la fois une épreuve et une véritable expérience qui me permit de découvrir de nouvelles manières de programmation et me donna de nouvelles compétences sur la visualisation dans l’espace. La principale difficulté fut que les orientations de la base de données, de l’affichage et des axes sont parfois différentes et pose parfois quelques problèmes de visualisation.

## **Format de donnée BDD**

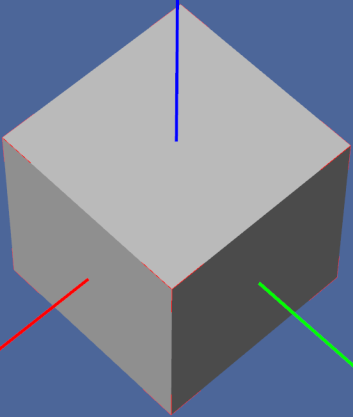
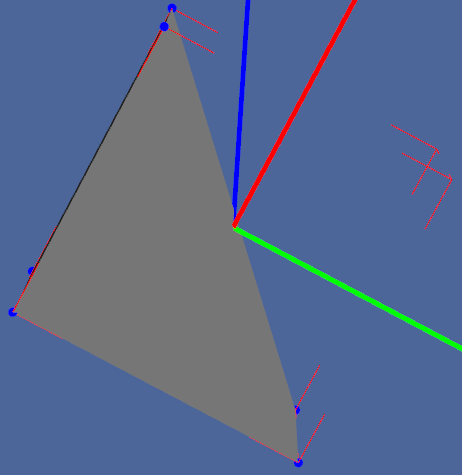
Le format de la base de données m’était inconnu et est très règlementé. Cependant après un certain temps je réussis facilement à m’y adapter. Malheureusement suite à une incompréhension personnelle, j’ai perdu une partie de mon temps à réparer mes erreurs : cette erreur fut sur les luminances et leurs relations avec les vecteurs et les sommets. Cependant, cette erreur repérée sur la fin, fut rapidement corrigée.

# Test et résultats

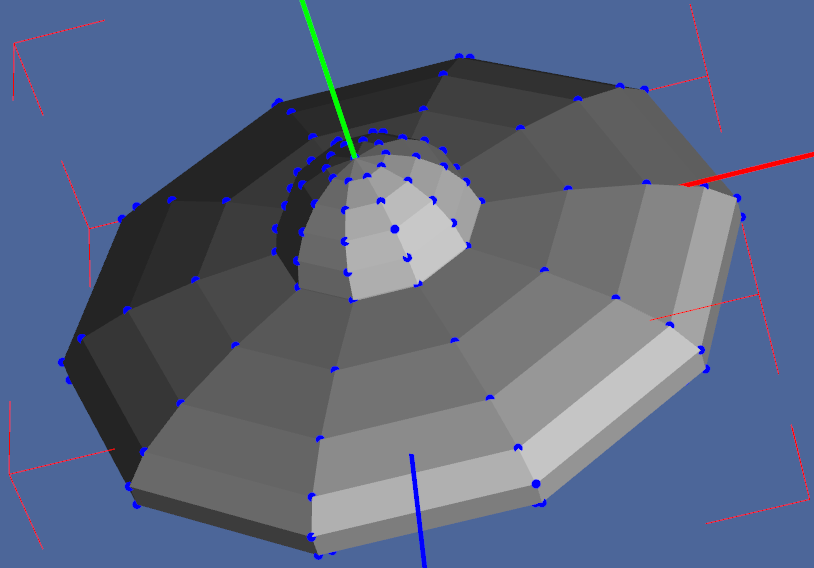
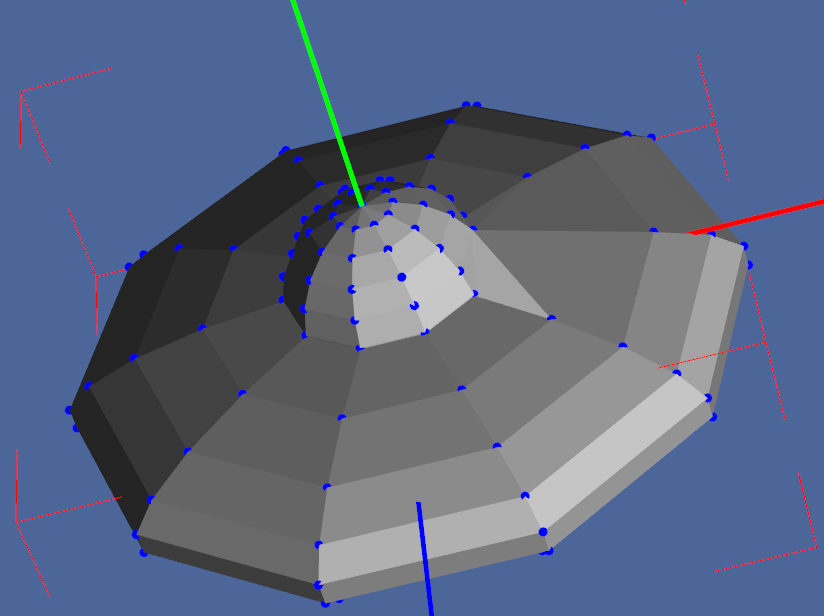
## **Test**

## Cube

Cette figure fut une des figures de proue pour les tests. Elle fut la principale figure testée (excepté la simplification et l’inversion des normales).



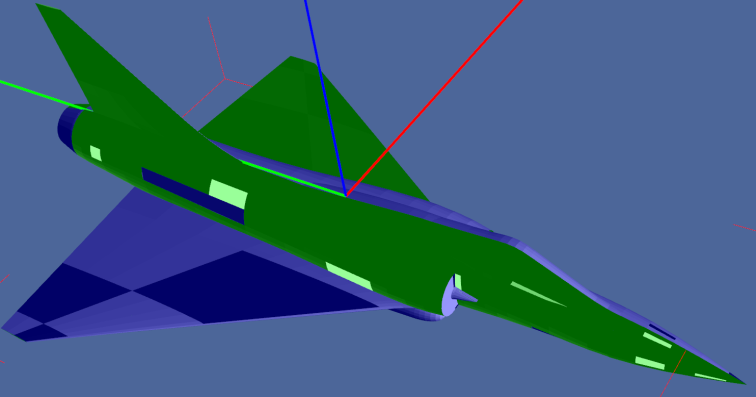
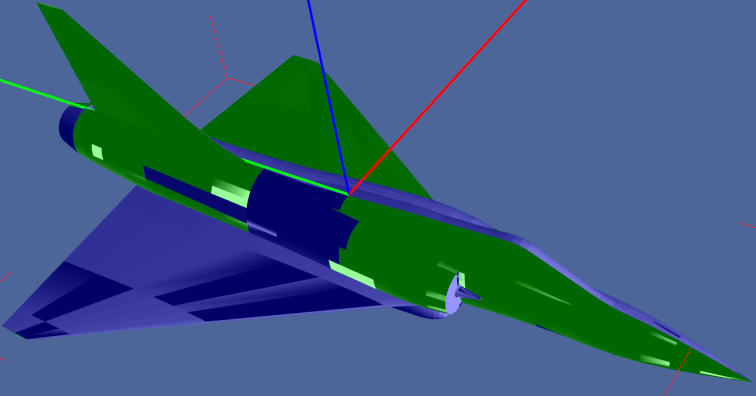
## OVNI

Cette figure fut aussi une des figures principales de test car elle possède 2 objets et non un seul comme dans le cube. Elle permit notamment de faire les tests de multi-objets et de soudure entre plusieurs objets.

## M2000

Cette figure permit 2 choses :

* tout d’abord faire du test multi-objets sur la simplification (car dans la base de données originale tous les sommets sont multiples) : on voit la différence, à gauche la base de données avec déjà le rajout des normales, luminances et vecteurs qui sont absent de la base de données originale et à droite la version avec inversion de plusieurs facettes au centre de l’image et une simplification de base de données où on voit la différences avec le lissage de Gouraud sur l’affichage.



## **Résultats**

Les nombreux tests se sont retrouvés concluants, malgré des bugs apparaissant suite à l’ajout de nouvelles fonctionnalités. Cependant malgré un début très rapide, les nombreuses difficultés survenues par la suite firent en sorte que je fus dans l’impossibilité de finir le projet car le temps fut trop court. Cependant la méthodologie appliquée feront en sorte que la/les personnes qui reprendront le projet pourront le récupérer facilement.

# Conclusion

Pour conclure ce projet j’aimerai parler de ma contribution envers l’ONERA, de l’avancement du projet et de ce que j’ai apporté personnellement dans ce projet à l’entreprise. Mais avant d’y arriver j’aimerai dire mon ressenti sur ce stage : ce stage m’a énormément fait réfléchir et m’a vraiment permit de développer l’idée que j’avais du suivi de projet grâce à l’expérience que j’ai eue l’année passée. Cela a fait en sorte que lorsque j’ai définit la méthodologie je voyais déjà les étapes suivantes et le fait que des personnes plus tard reprendraient mon travail. Mon objectif étant plus tard d’être chef de projet cela m’a vraiment permis de bâtir de bonnes bases pour ce futur objectif.

Maintenant parlons de ma contribution envers l’ONERA. Cette contribution se situe dans le fait que le logiciel OVNI, grâce à ce que j’ai développé, fera une très bonne base dans les années à venir pour les développements futurs.

Le fait d’avoir pris la décision de m’attaquer à un problème qui aurait dut être abordé dans un second temps, je pense, fera en sorte que le futur le projet se développera plus vite que si j’avais suivi la démarche proposée à l’origine. Cela permettra donc un gain de temps et d’argent et j’espère que ma méthodologie aura permis aux futurs contributeurs du projet d’avancer plus vite dans leur travail.

Le projet a d’ores et déjà une base bien développée. Il suffira de développer la majorité des fonctions annexes qui ajouteront des fonctionnalités au projet que ce soit au niveau de la lecture, de l’écriture, etc… Je pense qu’avec un ou deux stagiaires supplémentaires de 3 mois, sachant que les 2-3 premières semaines sont principalement là pour que le stagiaire comprenne le code, cela permettra d’obtenir un produit finit très intéressant.

Enfin, ce que j’ai apporté dans le projet et à l’entreprise, c’est principalement mon expérience personnelle du monde des projets et les multiples projets que j’ai eu à développer ou suivre au cours de mes études ou mes expériences passées. C’est pour cela que je ne regrette en rien d’avoir eu ce stage et donc, que je le referai sans hésiter. Mon seul regret aura été de ne pas pouvoir observer la continuité du développement et je souhaite bon courage au futur stagiaire.

# ANNEXE