Un FabLab est un atelier équipé de diverses machines pilotées par ordinateur, mises à disposition du public pour la conception et la réalisation d'objets. Par exemple, l'imprimante 3D permet d'« imprimer » des objets à partir de modèles 3D numériques ; la découpeuse laser permet de découper des formes complexes dans des plaques en bois, métal, plastique etc., à partir d'images vectorielles. Un des principes des FabLab est d'encourager le partage de ressources, y compris les ressources numériques tels les modèles d'objets 3D ou les images de modèles.

Il vous est demandé de réaliser un logiciel pour la gestion d'une bibliothèque de modèles 3D pour un FabLab. Les fonctionnalités minimales requises pour le logiciel sont :

— charger un fichier qui contient un modèle 3D et l'afficher à l'écran ;
— contrôler l'affichage du modèle (tourner, rapprocher) ;
— explorer une bibliothèque de modèles 3D.

Ce projet contribuera à vos notes de Projet tutoré (UE33), Modélisation mathématique (UE32), Conception objet avancée (UE31) et Production d'applications (UE31). Les évaluations pour chaque module sont séparées. Des informations sur les critères d'évaluation sont disponibles sur Moodle.

Le projet est découpé en deux parties (livrables) devant être rendues comme précisé sur Moodle. Les fonctionnalités demandées pour chacun des livrables sont décrites dans la suite. Vous trouverez un calendrier prévisionnel en annexe de ce document.

1 Livrable 1 : chargement et affichage d'un modèle 3D, bibliothèque de modèles

Un modèle 3D est donné par un ensemble de segments et de triangles dans un espace affine à trois dimensions. Les éléments du modèle sont bien identifiables sur l'image sur la Figure 1. Un modèle est décrit dans un fichier, suivant le format PLY, décrit brièvement sur Wikipedia : `https://en.wikipedia.org/wiki/PLY_(file_format)`. Des exemples de modèles 3D utilisant ce format se trouvent sur `http://graphics.stanford.edu/data/3Dscanrep/`.

Une bibliothèque de modèles est un répertoire qui contient des fichiers au format PLY.

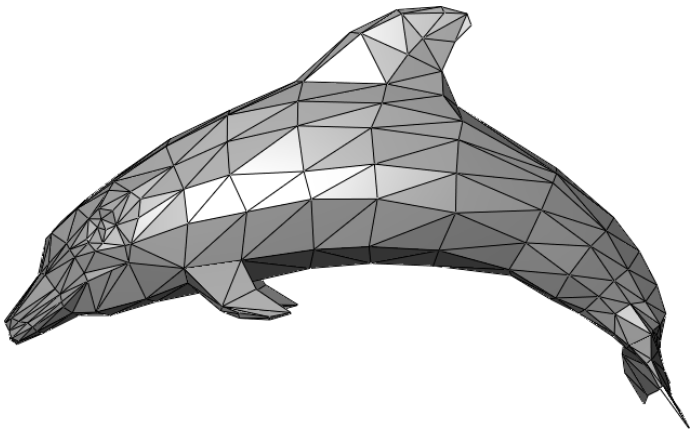


FIGURE 1 – Modèle 3D.

1.1 Programme exécutable

Pour le Livrable 1 vous devez faire un programme java exécutable (`.jar`) qui prend en charge une bibliothèque de modèles 3D depuis le nom du répertoire. Le programme affiche alors la liste des modèles disponibles, et propose à l'utilisateur d'en visualiser un. Si le fichier du modèle contient des erreurs (*cf* ci-dessous), le programme devra afficher un message d'erreur. Sinon, le programme doit visualiser (dans une fenêtre graphique) le modèle 3D de façon similaire à la Figure 2, en présentant trois vues de l'objet : de face, vue de haut et vue de côté [1].

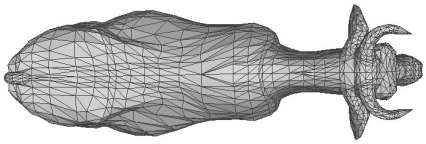
1. avec toutefois cette limitation que les faces visibles pourront être affichées avec une même teinte, sans calcul d'éclairage

Vous devrez permettre à l'utilisateur de contrôler l'affichage de l'objet 3D : tourner, zoomer, translater.

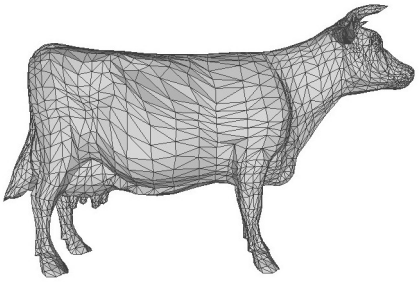
Les erreurs possibles dans le fichier sont : non respect du format, points ou segments manquants, définition de faces erronée, *etc.*

Fonctionnalités requises pour la librairie de modèles :

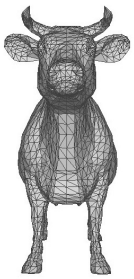
- afficher la liste des modèles, en donnant pour chacun : *nom de fichier*, *nombre de faces*, ainsi que les informations optionnelles qui pourraient être spécifiées dans le fichier à l'aide de commentaires en utilisant les mots clés `nom:`, `description:`, `dateCreation:` et `auteur:` ;
- permettre de choisir un modèle et de le visualiser.



(a) Vue secondaire « haut »



(b) Vue principale « face »



(c) Vue secondaire « droit »

FIGURE 2 – Affichage demandé pour le livrable 1.

Fonctionnalités requises pour l'affichage du modèle :

- trois vues du modèle sont visualisées simultanément : une vue principale de face et deux vues secondaires vu de la droite et vu de haut ;
- il est possible de transformer le modèle depuis la vue principale :
  - faire tourner,
  - changer d'échelle (agrandir, rétrécir)
  - déplacer droite/gauche, haut/bas.
- les deux vues secondaires changent en conséquence (voir exemple ci-dessous). Par exemple, si le modèle tourne, alors le côté « face » change, donc les côtés « droite » et « haut » changent également
- le contrôle s'effectue grâce à des boutons dans l'interface graphique (et optionnellement grâce à des raccourcis clavier et/ou grâce à la souris).

Exemple de transformation du modèle et comment cela affecte l'affichage. *À partir de l'affichage de la vache comme sur la Figure 2, imaginons cette suite d'actions.*

1. *On tourne la vache de 90 degrés sur l'axe des x de façon à la voir par le bas. Le côté « face » montre maintenant les pieds de la vache au premier plan. Par conséquent, le côté « haut » montre son flanc, et le côté « droit » montre la vache couchée regardant vers nous, avec sa tête vers la droite et ses pieds vers la gauche ;*
2. *On met à l'échelle* $1/2$*. La vache est donc deux fois plus petite sur les trois vues.*
3. *On déplace la vache vers le haut dans la vue principale. Elle est don déplacée vers le haut dans la vue « droit », et ne change pas dans la vue « haut ».*[2]

1.2 Tests unitaires

Vous devez écrire des tests unitaires pour toutes les classes qui effectuent des calculs mathématiques, et toutes les classes de gestion de la bibliothèque d'objets. Les classes de test doivent se trouver dans un dossier

2. On pourrait imaginer que dans la vue « haut » la vache se rapprocherait vers nous, mais ce n'est pas le cas car pour afficher vous aller faire une projection sans perspective dans laquelle les objets plus proches ne paraissent pas plus grands.

de code source séparé nommé `test`.

1.3 Documentations diverses

Vous devez ajouter quelques captures d'écran et une vidéo de 2-3 minutes qui présente votre projet ; à mettre dans le dossier prévu à cet effet.

1.4 À rendre

Le façon de rendre est décrite sur Moodle, et doit impérativement être respectée.

2 Livrable 2 : Contrôles et affichages avancés

La première exigence est d'organiser le programme selon le patron Modèle-Vue-Contrôleur (*cf* cours de Conception objet avancée), sans quoi les améliorations demandées dans la suite seraient très difficiles, voire impossibles à réaliser. Créez des *packages* séparés pour les vues, le modèle, et les contrôleurs. Chaque partie (par exemple le modèle) peut même avoir plusieurs *packages*. Organisez vos classes dans les différents *packages*. Cela peut nécessiter de décomposer certaines classes pour séparer les différentes préoccupations (modèle, vue, contrôleur).

2.1 Fonctionnalités requises

2.1.1 Affichage

L'affichage déjà implémenté doit autoriser en plus ces modes :
— affichage des faces seulement ;
— affichage des segments seulement ;
— affichage des segments et des faces (par défaut).

2.1.2 Gestion de la bibliothèque de modèles

Le contenu de la bibliothèque sera affiché dans la fenêtre de l'application en utilisant un tableau. Il est possible d'ouvrir un modèle depuis la bibliothèque, par exemple par double clic ou en utilisant un bouton. Il est possible de trier le tableau sur ses différentes colonnes.

2.2 Fonctionnalités au choix

Vous devez choisir lesquelles de ces fonctionnalités implémenter. Chacune peut vous apporter un nombre maximal de points en fonction de sa difficulté. Au total, 6 points sur 20 seront consacrés à ces fonctionnalités. Vous trouverez des informations détaillées sur Moodle.

2.2.1 Affichage

Modèle centré Un modèle est dit centré si on voit la totalité de l'objet (rien ne dépasse des bords), le centre de l'objet 3D occupe le centre de la vue, et le modèle occupe bien l'espace disponible (l'objet n'est pas tout petit au centre du panel). Lors du chargement et premier affichage d'un objet, celui-ci est centré dans les trois vues. De plus, l'utilisatrice peut à tout moment demander de centrer l'objet. Notez que dans ce cas le côté « face » actuel est gardé tel qu'actuellement.

Par exemple, revenons sur la suite de trois actions données en exemple dans la section précédente. Si au début la vache est centrée et après la troisième action on active la fonctionnalité centrer, la vache apparaitra toujours avec ses pieds au premier plan sur la vue principale, mais elle deviendra 2 fois plus grande, et elle sera déplacée à nouveau vers le bas.

Éclairage Pour l'instant l'affichage de modèles 3D se fait en coloriant chaque face (triangle) avec une couleur uniforme, ce qui donne un affichage 3D rudimentaire. On peut améliorer le résultat en définissant une source de lumière, qui est un point « à l'infini » dans l'espace ; vous prendrez un point fixé, par exemple à 45 degrés vers la droite de l'observateur. La couleur de chaque face dépend alors de l'angle entre les rayons qui proviennent de la source de lumière et le plan de la face.

Lissage Il s'agit d'une amélioration supplémentaire. Comme pour l'éclairage, il y a une source de lumière, mais la couleur d'une face n'est pas uniforme. Sur chaque face on affiche un gradient de couleur, qui dépend de l'éclairage des angles du triangle. Cette méthode permet d'avoir un affichage 3D de très

bonne qualité avec relativement peu de faces, mais elle est plus gourmande en calcul. À tester avec des objets avec peu de faces d'abord.

Ombre portée Fait de nouveau appel à une source lumineuse, mais cette fois-ci cette lumière va produire une ombre au « sol » du modèle 3D. Cette fonctionnalité est un peu plus simple que les précédentes, mais elle suppose quand même de définir le « sol » et la position du modèle 3D par rapport à celui-ci.

2.2.2 Gestion de la bibliothèque de modèles

— rechercher des modèles par mot clé en prenant en compte le nom, la description, le nom du fichier ;
— éditer les informations sur les modèles depuis l'interface graphique. Les changements seront alors enregistrés dans le fichier correspondant.

2.2.3 Autres

Vue en tranches Il s'agit d'une fonctionnalité utile dans un FabLab. Une technique souvent utilisée est de construire un objet en collant des tranches, comme sur la Figure 3.

FIGURE 3 – Objet construit en collant des tranches de bois.

Pour le faire, on doit pouvoir découper des tranches de la bonne forme. Vous devez implémenter le calcul de la forme des tranches.

Concrètement, l'utilisateur doit pouvoir choisir un modèle dans la bibliothèque et demander son découpage en un nombre de tranches qu'il va spécifier. Le découpage se fera suivant le plan parallèle au plan d'affichage, d'avant en arrière. Le logiciel devra alors afficher la première tranche, et permettre de naviguer entre les tranches en avant et en arrière.

Note : une tranche est définie par un ensemble de segments, qui sont les intersections des triangles du modèle 3D avec le plan de coupe.

Contrôleur horloge Le programme offre une option de rotation automatique du modèle 3D. Ceci s'implémente assez facilement à l'aide d'un contrôleur (MVC) automatique qui fera une rotation à chaque 1/2 seconde. Ce contrôleur requiert l'utilisation d'un *thread* séparé.

Autres Vous pouvez ajouter toute autre fonctionnalité qui vous semblent intéressante. L'enseignant·e de projet lui attribuera un nombre de points en fonction de sa difficulté.

2.3 Tests unitaires

Il faut ajouter des tests pour les nouvelles fonctionnalités qui requièrent des calcul mathématiques et celles liées à l manipulation de la bibliothèque d'objets.

2.4 Documentations diverses

Vous devez produire un **schéma UML**. Le schéma doit être *synthétique*, et aider à comprendre l'architecture du logiciel. Il doit contenir les classes principales, et seulement leurs méthodes principales, doit montrer les associations et leur donner un nom. En aucun cas le schéma UML ne doit être celui qu'on obtient par génération automatique à partir du code.

Vous devez également produire une documentation de vos classes au format Javadoc. Cette documentation doit être générée à l'aide de l'outil `javadoc` sur vos classes convenablement commentées.

Finalement, vous devez faire une vidéo de présentation des fonctionnalités de votre programme. La vidéo durera de **4 à 6 minutes** et sera faite par capture d'écran. Elle doit illustrer l'utilisation du programme d'affichage et montrera toutes les fonctionnalités du logiciel (affichage, manipulations de librairies de modèles, fonctionnalités supplémentaires). La vidéo peut être sous-titrée ou contenir des explications orales. Vous veillerez à présenter votre logiciel sous son meilleur jour, et illustrer ses performances par exemple en affichant des modèles volumineux et/ou des modèles de forme irrégulière difficiles à centrer.

2.5 À rendre

Le façon de rendre est décrite sur Moodle, et doit impérativement être respectée. Vous allez également faire une mini soutenance de votre projet devant vos enseignants impliqués dans le projet.

3 Objectifs, exigences, organisation du travail

Ce projet tuteuré vise à vous faire acquérir diverses compétences :

connaissances techniques programmation java, modélisation mathématique, utilisation de git ;

méthode de travail s'organiser, se coordonner, communiquer à l'intérieur de l'équipe et avec les enseignants ;

travail en autonomie lire et comprendre ce qui est demandé (ce document), utiliser internet pour se documenter (par ex. pour connaitre le format PLY).

La note finale essaie de sanctionner ces différents aspects.

D'après le programme du DUT, ce projet demande 50 heures de travail individuel par étudiant, soit plus de 3 heures par semaine en moyenne.

La quantité de travail de chacun·e sera sanctionnée individuellement : les étudiant·es ayant travaillé moins risquent d'avoir une moins bonne note. Attention tout de même : travailler beaucoup plus n'assure pas une meilleure note, au contraire. Le but est de travailler en équipe, il faut absolument éviter qu'un·e ou deux membres de l'équipe projet fassent tout le travail.

La contribution de chacun·e sera jugée d'après son activité sur le dépôt git.
Il est donc primordial de bien configurer git sur toutes les machines d'où vous faites des commits.

Toute contribution sera prise en compte (test, documentation, faire la vidéo). Cependant, un·e étudiant·e ne peut pas se consacrer uniquement à des tâches qui ne nécessitent pas de programmer. Chacun·e doit contribuer en écrivant du code original.

Si vous avez contribué avec quelque chose qui n'est pas visible dans *git* (par ex. recherche de modèles 3D, capture de la vidéo de présentation, préparation de l'archive à rendre, etc.), assurez vous que c'est mentionné dans le rapport.

En cas de conflits au sein de l'équipe, en parler à l'enseignant·e de TP au plus vite ; il/elle pourra jouer le rôle de médiateur pour vous aider à résoudre le problème.

Fonctionnalités requises, optionnelles et supplémentaires. Vouloir personnaliser son projet, c'est très bien. Vous pouvez vous faire plaisir et ajouter toutes les fonctionnalités supplémentaires que vous voulez. Sachez cependant que votre note finale dépendra en grande partie des fonctionnalités obligatoires, et seulement à la marge des améliorations personnelles.