**1. Sommaire**

À ce stade, l’application consiste encore en une collection de fonctions et de modules qui seront utilisés pour le travail du TP2. Le logiciel sera un croisement entre une application de dessin, un éditeur de textures et de cartes UV, ainsi qu’un visualisateur de modèles 3D.

Le but est de pouvoir modifier l’apparence de modèles 3D avec l’éditeur de textures, qui peuvent être exportées en fichier image, de manière rapide, simple et efficace. Des shaders et cartes de texture (height map, metallique, etc.) peuvent être importés et appliqués.

Les textures et matériaux produits pourront être visualisés sur plusieurs types de primitives 3D (sphère, cube, plan, capsule, etc.), ainsi que sur un modèle au format compatible importé par l’utilisateur.

Les modules et fonctionnalités implémentées pour cette première partie du travail consistent surtout des fonctionnalités de dessin, d’import-export de fichiers images et 3D, d’une base solide pour l’interface utilisateur et de quelques critères fonctionnels qui n’ont pas de réelle utilité, mais qui sont implémentés pour fins de démonstration/prérequis.

À terme, l’application pourra servir pour avoir un aperçu de l’apparence de modèles 3D, et d’y apporter des modifications, si nécessaire. Il est à noter que la géométrie des modèles ne sera pas modifiable en soi.

**2. Interactivité**

Il est possible d’importer tous les principaux types de fichiers images (bmp, png, jpg, jpeg) dans l’application avec un simple glisser-déposer. Les images seront automatiquement chargées et dessinées.

La manière la plus simple d’importer un modèle 3D est simplement d’effectuer un glisser-déposer du fichier dans la fenêtre de l’application. Tout comme les fichiers images, les modèles 3D seront automatiquement importés dans la scène à l’endroit où ils ont été déposés.

Il est possible d’importer plusieurs modèles et images simultanément, mais leurs postions devront ensuite être réajustées, car tous les fichiers auront le même point d’origine lors du glissé-déposé.

Le raccourci clavier ‘Shift+S’ peut être utilisé pour exporter un rendu de la fenêtre de l’application qui sera sauvegardé dans le répertoire du projet.

La plupart des actions possibles sont disponible à la fois via le menu de l’interface graphique, soit via un raccourci-clavier. Par exemple :

* La touche < . > (point) permet de passer rapidement d’une projection orthogonale vers une projection en perspective, et vice-versa.
* La touche < z > active et désactive la rotation automatique des modèles 3D affichés.
* La touche < x > active et désactive une animation de ‘flottement’ (*wobbling*) sur les modèles 3D.
* La touche < g > active et désactive l’affichage de l’interface utilisateur.
* Le déroulement de la molette de la souris permet de faire un zoom de la caméra.
* Les touches 0, 2, 4, 6 et 8 du clavier *numpad* permettent de positionner la caméra à des positions prédéfinies.

**3. Technologie**

Le projet est basé sur la librairie OpenFrameworks qui est compatible sur une multitude de plateformes. Le code a été créé en parallèle dans :

* Visual Studio 2022
* Xcode

Plusieurs applications tierces ont cependant été utilisées pour des tâches de production, comme :

* Blender et 3DS Max : création et la conversion de modèles 3D.
* Inkscape : création de fichiers .SVG
* GIMP : création de fichiers images (bmp, png, jpg, jpeg, etc.)

Un contrôle source a été implémenté à l’aide de :

* GitHub
* GitHub Desktop

Bien entendu, l’application est basée sur la technologie OpenGL pour effectuer son rendu dans la fenêtre d’affichage.

**4. Compilation**

L’architecture du programme est extrêmement simple et n’utilise aucune dépendance externe à OpenFrameworks, il est donc possible de compiler sur toutes les plateformes compatibles.

Cependant, certains modules doivent être présent dans le projet :

* ofxGUI
* ofxAssimpModelLoader
* ofxSVG

La méthode la plus simple pour la compilation est d’utiliser le ProjectGenerator d’OpenFrameworks et de créer un projet en incluant les modules mentionnés ci-dessus.

Une fois le projet généré dans votre IDE de choix (recommandé : Xcode ou Visual Studio 2022), il suffit de remplacer les fichiers suivant par ceux fournis dans l’archive .zip :

* main.cpp
* ofApp.h
* ofApp.cpp

Le projet utilise le module ofxGUI pour l’interface utilisateur, qui possède une police de caractère par défaut qui est très peu agréable à l’œil. C’est pourquoi une police de caractère est fournie dans le fichier ‘bin’ de l’archive. Ce fichier peut simplement être copié dans le répertoire du projet.

Le fichier ‘bin’ contient également un modèle 3D d’un mannequin humain, qui peut être utilisé pour tester les fonctions du logiciel.

Une machine compatible avec la version 3.3+ de OpenGL est requise pour lancer l’application. Il est également à noter que l’application n’a pas été testée sur une architecture 32 bits et pourrait exhiber des comportements inattendus sur de telles architectures.

**5. Architecture**

L’architecture de l’application, à ce stade, est d’une simplicité absolue : tout le code est contenu dans la classe ofApp que génère le ProjectGenerator de OpenFrameworks.

Une utilisation maximale des fonctions OpenFrameworks permet toutefois de simplifier la quantité de code nécessaire au bon fonctionnement de l’application. Cependant, des structures de données faites sur mesure ont été implémentés afin de simplifier le fonctionnement de l’application et ainsi que la lecture et la compréhension du code.

Notamment, plusieurs ‘structs’ ont été définis pour différents éléments, comme les primitives vectorielles et les objets 3D. Lorsqu’un nouvel élément est créé ou importé dans l’application, une nouvelle structure de donnée est créée dépendamment du type de ressource dont il s’agit. Cette structure enregistrera tous les paramètres comme la position, la dimension, la couleur de remplissage et de contour, etc., en plus des données brutes de la ressource elle-même.

Les éléments crées sont ensuite poussés dans un vecteur qui constitue essentiellement la hiérarchie de la scène.

Cette méthode permet d’ajouter, retirer et modifier les différents éléments de façon dynamique lors de l’utilisation de l’application et de simplifier grandement la lecture et l’écriture du code.

Une simple itération de ces vecteurs de hiérarchie permet ainsi de facilement dessiner tous les éléments dynamiques de la scène.

**6. Fonctionnalités**

* Critère 1.1 – Importation d’images
  + Il est possible d’importer des images dans tous les principaux formats avec un simple glissé-déposé d’un fichier image sur la fenêtre d’affichage. Si le format est valide et l’image n’est pas corrompue, l’image sera ajoutée è la hiérarchie de la scène et affichée à la position où le fichier à été déposé.
  + Formats d’image supportés :
    - .bmp
    - .png
    - .jpg
    - .jpeg
  + Il est possible d’importer une multi-sélection de fichiers en une seule opération.
* Critère 1.2 – Exportation d’images
  + Il est possible d’exporter un rendu de la fenêtre d’affichage en utilisant la combinaison ‘Shift+S’
  + Localisation de l’image:
    - Répertoire du projet
  + Nomenclature de l’image exportée :
    - ‘screenshot\_XYZ’, où XYZ est un index unique.
  + Formats de l’image exportée :
    - .jpeg
* **Critère 2.2 – Outils de dessin**
  + Le panneau ‘’ permet de modifier les paramètres de dessin, comme la couleur, le remplissage et le contour.
    - Ces paramètres sont ensuite enregistrés dans le struct de l’élément dessiné.
  + Paramètres :
    - ‘Activer contour, activer remplissage, modifier la couleur de contour et de remplissage, modifier la largeur du contour.
  + La couleur de l’arrière-plan est également modifiable, mais elle n’est pas passée aux éléments de dessin lors de leur création.
* **Critère 2.3 – Primitives vectorielles**
  + Le panneau ‘Paint tools’ (à droite de l’écran) permet de sélectionner la primitive vectorielle désirée.
  + Primitives disponibles :
    - Ligne, cercle, ellipse, triangle, rectangle/carré, ainsi qu’un polygone sur-mesure.
  + Il suffit de cliquer sur le bouton du panneau correspondant à la primitive choisie pour la sélectionner.
* **Critère 2.5 – Interface**
  + Le panneau ‘Paint tools’ (à droite de l’écran) permet de sélectionner la primitive vectorielle désirée.
  + Primitives disponibles :
    - Ligne, cercle, ellipse, triangle, rectangle/carré, ainsi qu’un polygone sur-mesure.
  + Il suffit de cliquer sur le bouton du panneau correspondant à la primitive choisie pour la sélectionner.
* **Critère 3.1 – Graphe de scène**
  + Le panneau ‘Scene Hierarchy’ (à droite de l’écran) permet de visualiser et de sélectionner les éléments présents de la scène.
  + Il s’agit simplement d’un outil de visualisation et de sélection : la hiérarchie parent-enfant n’est pas supportée dans la hiérarchie.
* **Critère 4.3 – Modèles 3D**
  + L’application permet d’importer des modèles 3D via glisser-déposer d’un fichier dans la fenêtre d’affichage.
  + La multi-sélection est possible afin d’importer plusieurs modèles en une seule action.
  + Les éléments sont automatiquement ajoutés è la hiérarchie de la scène.
* **Critère 4.4– Animation**
  + L’application permet d’animer les modèles 3D de la scène afin de mieux visualiser les modèles.
  + La rotation et le *‘wobble*’ sont disponibles.
    - Activer/désactiver la rotation : Touche clavier ‘Z’
    - Activer/désactiver le wobble : Touche clavier ‘X’
* **Critère 5.1– Point de vue**
  + Il est possible de déplacer la caméra avec la molette de la souris
    - Pan : Enfoncer la molette et déplacez la souris
    - Zoom : Roulez la molette de la souris
  + La caméra peut être positionnée à des positions prédéfinies grâce aux touches du ‘numpad’ :
    - 0, 2, 4, 6 et 8
* **Critère 5.2– Projection**
  + La touche <.> (le point) permet d’alterner entre un mode de vue en projection et un mode de vue orthographique.
  + Les points de vue de caméra assignés au *numpad* (voir critère 5.1) ajustent automatiquement leur point de vue lorsque nécessaire.

**7. Ressources**

Très peu de ressources externes ont été nécessaires pour ce projet. Un seul modèle 3D est fourni dans le projet, afin de permettre de tester les fonctionnalités de l’application. Ce modèle de mannequin à été créé par moi-même, Simon Taddio, dans le cadre du cours de l’Université Laval *ANI-1702 Modélisation et Animation 3D.*

Certaines parties du code, comme le shader Lambert par défaut, ont été tirées des laboratoires et démonstrations du cours de l’Université Laval *IFT-3100 Infographie,* disponible sur GitHub1.

La police de caractère incluse, ‘RobertoSans-Regular’ provient du site 1001 Fonts2, mais son créateur est anonyme et la licence incluse ne mentionne pas son nom.

Le reste des ressources créés dans le cadre de ce travail consiste en les éléments du livrable, soit une vidéo de présentation/démo, un formulaire d’autoévaluation, ainsi que ce présent document de conception.

1https://github.com/philvoyer/IFT3100H23

2http://www.1001fonts.com/roberto-sans-font.html

**8. Présentation**

Le projet a été réalisé individuellement. L’ensemble du code, de l’architecture, de la conception et des ressources originales ont été faites par Simon Taddio.

Quelques ressources externes ont été utilisées pour fins de démonstration dans la vidéo de présentation (images, modèles 3D, etc.), mais ne sont pas incluses dans le projet, ni nécessaires au bon fonctionnement de l’application.

Le fichier ‘bin’ contient les éléments suivants :

* La police de caractère de l’interface utilisateur provient de FOOBAR.COM et est disponible sous la licence CC!@$.
* Le modèle 3D ‘mannequin.obj’ a été réalisé par Simon Taddio.