Document de design

1. Sommaire

Produit dans le cadre du cours IFT3100, le présent document de design accompagne une application développée en tant que projet de session. La présente mouture est présentée en tant que travail final, englobant la partie déjà évaluée et présentée à la mi-session et le travail fait en lien avec la deuxième partie du cours.

L'application demandée a pour objectif de construire, éditer et rendre des scènes visuelles. Dans le cadre du présent livrable, nous nous sommes concentré sur les aspects vues en deuxième moitié de session, soit les caméras, l'illumination classique, le lancer de rayons, la topologie et l'illumination moderne.

Le présent document contient des informations sur le développement de l'application, les technologies utilisées, les dépendances nécessaires à son utilisation et son architecture ainsi qu'un guide d'utilisation de l'application et des détails sur les différentes fonctionnalités implantées.

Dans le présent document, une lettre ou un chiffre entre parenthèses *(a)* représente une touche clavier associée à une fonctionnalité. Une liste complète des raccourcis clavier est aussi disponible à la section 2 du présent document.

2. *Interactivité*

En tout temps, il est possible d'exporter une capture d'écran en appuyant sur la touche appropriée *(1).* L'image produite sera du format *.png*.

En tout temps, il est possible d'importer une image pour analyse dans l'histogramme en la déposant *(drag and drop)* dans la fenêtre de l'application. L'image remplacera alors l'image en cours et pourra être visible en appuyant sur la touche appropriée *(i).*

Liste des touches et de leurs fonctionnalités

0	Basculer entre le mode de projection orthogonal et perspective, en mode caméra seulement (v)
1	Capturer une image *(.png)*
7	Activer/désactiver la lumière directionnelle, en mode illumination activée (l)
8	Activer/désactiver la lumière ponctuelle, en mode illumination activée (l)
9	Activer/désactiver la lumière projecteur, en mode illumination activée (l)
a	Activer/désactiver le shader de tessellation pour les formes 3D
b	Afficher/cacher les boîtes de délimitation
c	Afficher/cacher le cubemap
f	Changer le filtre appliqué sur l'image pour l'histogramme *(Identité – Emboss – Sharpen - Blur)*
g	Afficher/cacher la grille sur le plan xy
h	Afficher/cacher l'histogramme
i	Afficher/cacher l'image pour l'histogramme
l	Activer/Désactiver l'illumination de la scène
m	Changer le mode d'affichage des modèles 3D *(Sommets - Maillage - Solide)*
r	Remise à zéro
t	Afficher/cacher les textures de modèles 3D
u	Afficher/cacher l'interface

v	Activer/désactiver le mode caméra. Active du même coup l'illumination de la scène
x	Changer le matériau des modèles 3D *(Aucun - Cuivre - Chrome - Plastique - Caoutchouc)*
y	Basculer entre le mode shader et le mode matériau pour les modèles 3D
z	Changer le shader actif des modèles 3D *(Color fill - PBR - Lambert - Gouraud - Phong - Blinn-Phong)*
Backspace	Supprimer une forme vectorielle sélectionnée

3. Technologie

L'application a été développée en *C++* avec la librairie *OpenFrameworks*, sur *macOs/XCode*. En plus de cette librairie, certains *addons* ont été utilisés pour des fonctionnalités précises.

OpenFrameworks est une librairie permettant l'utilisation d'OpenGL de façon simple et intégrée.

https://openframeworks.com/

ofxGui est un *addon* permettant d'ajouter une interface graphique interactive aux projets OpenFrameworks. Toute l'interface de l'application est rendue par cet outil.

https://openframeworks.cc/documentation/ofxGui/

ofxGrafica est un *addon* permettant d'ajouter des graphiques à OpenFrameworks. Il a été utilisé spécifiquement pour ajouter à l'application un histogramme à trois niveaux dans le même graphique.

https://github.com/jagracar/ofxGrafica

ofxAssimpModelLoader est un *addon* permettant l'importation simple de modèles 3D et leur traitement pour ensuite être intégrés à OpenFrameworks.

https://openframeworks.cc/documentation/ofxAssimpModelLoader
/ofxAssimpModelLoader/

ofxRaycaster est un addon permettant l'implémentation simple d'un lancer de rayon en fournissant notamment des méthodes pour le calcul d'intersections et de rayons incidents.

https://github.com/edap/ofxRaycaster

ofxReflectionRefraction est un addon qui génère un effet de réfraction, utilisé dans le cubemap.

https://github.com/kashimAstro/ofxReflectionRefraction

4. Compilation

Pour compiler le programme, il est primordial de tout d'abord ajouter les *addon* ofxReflectionRefraction et ofxRaycaster, fournis avec ce livrable, ainsi que de s'assurer d'avoir les *addons* ofxGrafica, ofxAssimpModelLoader et ofxGui présents dans le dossier source d'OpenFrameworks. Il est alors simple de préparer le code source pour la compilation grâce à l'outil ProjectGenerator fourni par OpenFrameworks. Une fois la mise à jour du projet effectuée, en s'assurant de spécifier les addons nécessaires nommés plus haut, le projet peut être compilé sur Xcode et lancé par ce même outil.

5. Architecture

L'architecture du logiciel est principalement calquée sur le modèle d'OpenFrameworks, tel que présenté dans le cadre du cours. Le fichier *main.cpp* sert de base de lancement du programme, qui appelle ensuite l'application en soit.

L'application est gérée par *application.cpp*. On retrouve dans ce fichier source tout le code concernant l'interface graphique et les fonctions relatives aux mouvements de souris et au clavier. Ce fichier sert aussi de lien entre l'utilisateur et le fichier de rendu. En effet, chaque ajout d'une nouvelle primitive ou d'un nouveau modèle et chaque transformation est à la base traitée par *application.cpp* qui envoi ensuite la commande à *renderer.cpp*, selon l'outil sélectionné.

Le rendu est le rôle de *renderer.cpp*, appelé par *application.cpp* par une commande de dessin *draw*. Cette commande prend toutes les actions effectuées et fait le rendu selon les spécifications de chaque primitive ou modèle. Le *draw* est répété en boucle et assure donc tout le rendu. Selon les spécifications choisies, un shader peut aussi être appelé durant la séquence *draw* et appliqué sur les modèles 3D, à l'exception de la banane qui a sa propre texture.

Plusieurs calculs se font aussi au niveau de la commande *update*, qui elle aussi roule en boucle. C'est notamment là que s'effectue les mises à jours des lumières selon les configurations choisies, que le calcul des courbes Catmull-Rom et des surfaces de Coons sont effectués et que les différents shaders sont chargés, au besoin.

6. Fonctionnalités

6.1 Caméra

6.1.1 Point de vue

Il est possible de transformer une caméra par rapport à une ou des entités géométriques et l'utilisateur peut manipuler interactivement la caméra pour voir la position centrale de la sélection de différents points de vue, en plus de pouvoir s'en approcher et s'en éloigner.

Le mode caméra (v) permet à l'utilisateur de tourner en cliquant et bougeant (*click and drag)* et de s'approcher et s'éloigner en utilisant le bouton central de la souris. Il est aussi possible de limiter la rotation en cliquant sur les contours de la fenêtre puis en bougeant. Plusieurs fonctionnalités sont exclusives au mode caméra et seront indiquées comme telles.

6.1.2 Mode de projection

Au moins deux types de mode de projection (ex. perspective et orthogonale) sont supportés par l'application et offrent à l'utilisateur un point de vue cohérent de la scène.

Il est possible de basculer du mode de projection orthogonale au mode perspective et vice-versa en utilisant la touche (0) (zéro). Cette fonctionnalité est bien évidemment seulement visible en mode caméra.

6.2 Illumination Classique

6.2.1 Modèles d'illumination

Le rendu d'au moins 3 éléments visuels est fait à partir d'au moins 3 modèles d'illumination différents (ex. Lambert, Gouraud, Phong, Blinn-Phong ou autre).

Il est possible de visualiser les modèles d'illumination Lambert, Gouraud, Phong et Blinn-Phong en basculant en mode *shader* (y), puis en changeant de shader avec (z). Pour des résultats optimaux, le modèles 3D du ballon de soccer est recommandé. Les shaders sont seulement appliqués en mode caméra (v).

6.2.2 Matériaux

Au moins 3 éléments visuels d'une scène ont une surface avec un matériau sélectionné parmi un ensemble d'au moins 3 matériaux différents.

Il est possible de visualiser 4 matériaux différents sur les modèles 3D, en basculant en mode *material* (y), puis en changeant de matériau avec (x). Les matériaux sont seulement visibles en mode caméra (v).

6.2.3 Types de lumières

Il est possible d'avoir au moins une instance de 4 types de lumières différents (ambiante, directionnelle, ponctuelle, projecteur ou autre).

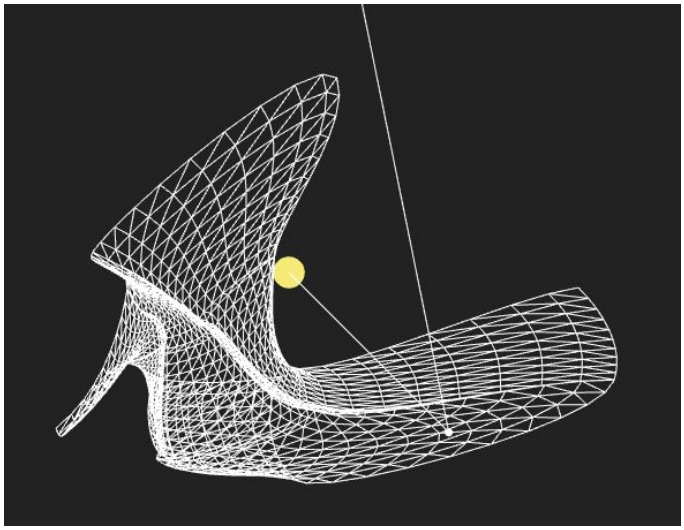
Une lumière ambiante est toujours présente lorsque le mode illumination (l) est activé. Il est possible de modifier sa couleur et sa position à l'aide de l'interface. De plus, des lumières de types directionnelle, ponctuelle et projecteur sont disponibles. Similairement, il est possible de configurer leur position et couleur, ainsi que leur orientation lorsque pertinent. Les shaders ne prennent

pas en compte les lumières autres que ambiante donc il est nécessaire de visualiser un modèle en mode matériau (y) (x) pour bien voir l'effet des lumières. De plus, il est possible d'activer et désactiver les lumières directionnelle (7), ponctuelle (8) et projecteur (9) avec des raccourcis claviers pour faciliter la visualisation. Les lumières peuvent être visualisées en tout temps mais devraient être visualisées en mode caméra (v) pour permettre une rotation autour des modèles.

6.3 Lancer de rayon

6.3.1 Intersection

L'application est capable de calculer l'intersection entre un rayon et au moins 3 types de primitives géométriques.



Il est possible de générer un rayon et d'afficher le rayon, l'intersection et la réflection du rayon sur la surface pour les primitives géométriques de type 2D (2D Draw), de type 3D (3D Draw) et pour la surface de Coons générée selon les courbes Catmull-Rom (voir section 6.4.2). Deux rayons distincts sont générés, un pour le 3D et un pour le 2D. Les deux rayons sont contrôlés par le même interface mais composante de l'axe Z est ignoré pour les surfaces 2D. Le rayon est seulement visible en mode caméra (v).

6.3.2 Réfraction

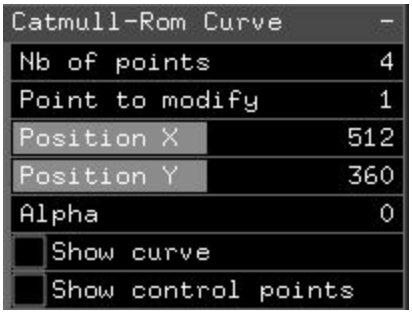
Une technique de rendu inspirée des principes du lancer de rayon est utilisée pour rendre au moins 1 effet de réfraction. (ex. une surface en verre)

Il est possible de voir la réfraction en ouvrant le cubemap (c). La réfraction fonctionne bien en regardant devant lors de l'ouverture mais se détériore lorsqu'un tourne la caméra vers l'arrière. Néanmoins, l'effet est bien visible en regardant le bâtiment et l'arc en ciel à travers la sphère vitrée. Le cubemap est seulement visible en mode caméra (v).

6.4 Topologie

6.4.1 Courbe paramétrique

Il est possible de rendre au moins 1 type de courbe paramétrique avec plus de 4 points de contrôle, par exemple une spline de Bézier ou de Catmull-Rom.

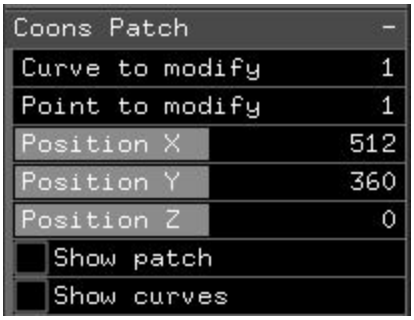


Le modèle Catmull-Rom a été choisi comme courbe paramétrique. Pour ajuster la courbe, l'utilisateur doit utiliser la section *Catmull-Rom Curve* de

l'interface, choisir le nombre de points de contrôles voulus (minimum 4 points, incluant les deux points externes), et ensuite sélectionner manuellement les valeurs de chaque point de contrôle et utilisant la deuxième barre (Point to modify). L'utilisateur pour aussi modifier la valeur du *alpha*, passant de la version classique dites uniforme à la valeur 0 à la version chordale à la valeur 1. Les calculs pour la courbe sont basés sur l'approche pyramidale de Barry et Goldman[1]. L'utilisateur peut finalement choisir d'afficher ou non les points de contrôles pour faciliter la visualisation.

6.4.2 Surface paramétrique

Il est possible de rendre au moins 1 surface paramétrique, par exemple une surface de Bézier bicubique ou une surface de Coons.



La surface de Coons a été choisie comme surface paramétrique. Elle est constituée de quatre courbes de Catmull-Rom à 6 points de contrôles chacunes, permettant beaucoup de variation dans la surface. L'utilisateur doit tout d'abord sélectionner une courbe à modifier (*Curve to modify),* puis un point à modifier sur cette courbe. Certains points sont reliés pour satisfaire le modèle et les mises à jour seront faites automatiquement. Ainsi, si le point 2 de la courbe 1 est modifié, le point 2 de la courbe 3 le sera du même coup. Une fois la surface affichée, l'utilisateur peut toujours modifier les valeurs des courbes et voir les changement en temps réel. De plus, il est possible de voir l'intersection avec un rayon si l'option *Show ray* est activée dans la section *Ray casting*. La surface peut être visualisée comme un

[1] https://en.wikipedia.org/wiki/Centripetal_Catmull%E2%80%93Rom_spline#Definition

maillage, un ensemble de points ou une surface pleine en changeant le mode d'affichage (m).

6.4.3 Shader de tesselation

Un shader de tesselation permet de sous-diviser la géométrie d'au moins 1 modèle, par exemple un plan ou une surface paramétrique.

Un shader de tesselation peut être appliqué sur les formes 3D (3D Draw) (a). Un bug reste présent dans une partie des formes mais la tesselation s'effectue correctement.

6.5 Illumination moderne

6.5.1 Métallicité

Au moins un matériau utilise un facteur de métallicité pour simuler le niveau de réflexion entre diélectrique et métallique.

Il est possible de visualiser un matériau qui utilise en facteur de métallicité pour simuler la réflexion. Le mode shader (y) PBR doit être sélectionné (z). Pour un meilleur rendu, il est recommandé d'utiliser le modèle de ballon de soccer.

6.5.2 Microfacettes

Au moins un matériau utilise avec des microfacettes pour simuler la rugosité de sa surface.

Il est possible de visualiser un matériau qui des microfacettes pour simuler la rugosité. Le mode shader (y) PBR doit être sélectionné (z). Pour un meilleur rendu, il est recommandé d'utiliser le modèle de ballon de soccer.

7. Ressources

Dans le cadre de ce livrable, toutes les ressources présentes ont trouvées sur Internet, de sources offrant du contenu libre de droits.

Ressources pour la deuxième partie du travail

Cubemap pour réfraction :
http://www.humus.name/index.php?page=Cubemap&item=Rainbow

Shader de tessellation
https://github.com/leozimmerman/ShadersLibrary

Configuration des matériaux
http://www.it.hiof.no/~borres/j3d/explain/light/p-materials.html

Textures module 10
https://texturehaven.com/

Ressources pour la première partie du travail

Image de base pour histogramme:
https://www.pexels.com/photo/person-walking-on-street-1981945/

Cubemap :
http://www.custommapmakers.org/skyboxes/zips/ame_nebula.zip

Modèles 3D

Banane :
https://www.turbosquid.com/3d-models/free-obj-mode-banana-fruit/682101

Cœur :
https://www.turbosquid.com/3d-models/free-love-heart-3d-model/1110350

Baymax :
https://www.turbosquid.com/3d-models/free-max-mode-baymax-big-hero-6/881191

Ballon de soccer :
https://www.turbosquid.com/3d-models/classic-football-ball-3d-model-1290297

De plus, certains algorithmes trouvés en ligne ont été la source d'inspiration pour la création des formes 3D.

Cube :
http://ilkinulas.github.io/development/unity/2016/04/30/cube-mesh-in-unity3d.html

Polyèdre 20 faces :
https://wiki.mcneel.com/developer/scriptsamples/icosahedron

8. Présentation

Le travail présenté a été réalisé seul, par Olivier Léveillé-Gauvin, étudiant au baccalauréat en informatique à l'Université Laval en dernière année. Ce travail représente les premiers pas de ce dernier dans le domaine de l'infographie. Une grande période d'apprentissage a donc initialement été nécessaire. Toutefois, une fois les bases acquises, ce projet s'est avéré très intéressant et, malgré la charge de travail que représente ce travail pour une seule personne le résultat final est satisfaisant, bien que pas aussi complet que voulu.

En effet, il a été impossible de terminer le travail de façon aussi complète que voulu, notamment l'arrivée d'un deuxième enfant au courant de la fin de session. Cet événement a eu pour effet de ralentir la cadence de travail, ce qui a malheureusement un impact sur le résultat final. Néanmoins, le travail accompli reste satisfaisant et restera sans aucun doute comme un des travaux universitaires les plus marquants. Bien au delà du résultat final, le cadre de ce travail a permis d'apprendre de nouvelles choses, notamment la générations de surfaces 3D réactives au courbes qui la délimitent, et ces apprentissages resteront précieux.