DÉPARTEMENT D'INFORMATIQUE
Université Laval

Travail Pratique #2
Document de design

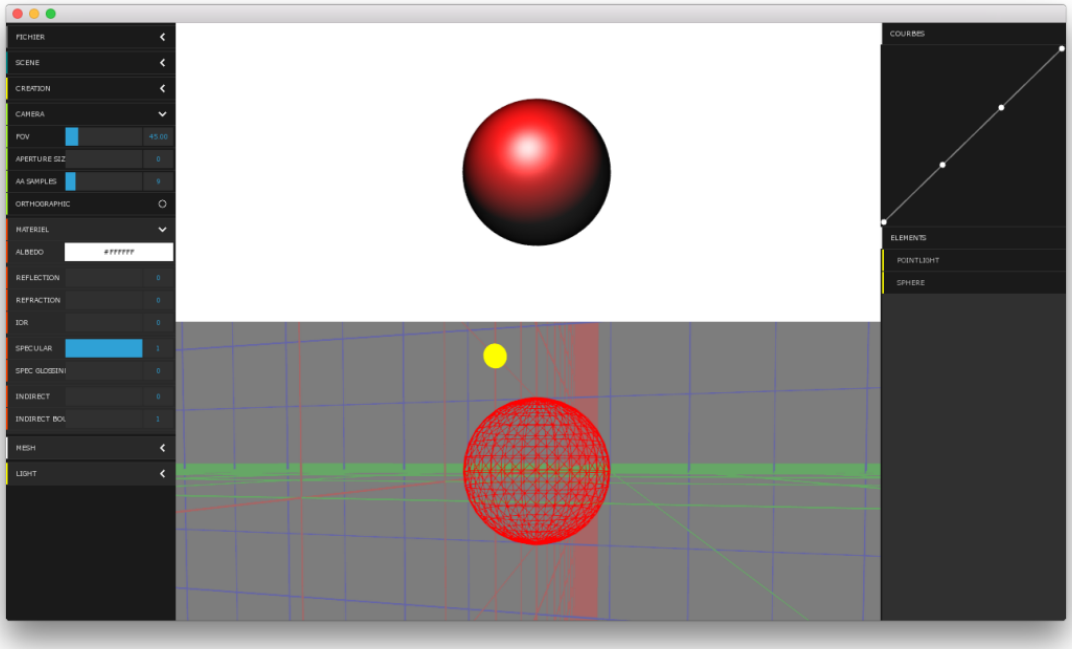
PRÉSENTÉ PAR
Yohan Poirier-Ginter
Vincent Côté-Giroux
Michaël Dodier

Infographie
IFT-3100-H18

30 Avril 2018

#1 Sommaire :

Le but premier d’application est d'explorer les techniques de rendu par raytracing, et donc l'emphase a été mise sur les fonctionnalités de rendu. L'engin de raytracing a été implémenté au niveau CPU, pour permettre plus de facilité aux niveaux de l'ajout d'effets complexes comme l'illumination globale. Une interface minimale a été incluse pour permettre l'exploration des différents paramètres du moteur de rendu. L'application a été construire à l'aide d'Openframeworks pour l'affichage d'un viewport ainsi que de l'interface mais la majorité de l'engin de rendu a été programmé de façon indépendante, incluant les effets de cameras, les matériaux, les collisions, les primitives géométriques et le loop de rendu. Le but du projet est de faire une preuve de concept avec des fonctionnalités semblables à celle du programme “Blender”. On parle ici d'être capable de faire du rendu 3D supportant des effets de raytracing, ainsi que la manipulation d'objets et de matériaux. Donc pour faire simple, notre application consiste à permettre la création et l'édition d'image 3D et de « mesh » à l'aide de l'interface et d'une panoplie d'outils de dessin qui peuvent entre autres afficher des surfaces paramétriques et gérer des lumières. Les lumières sur la scène et leurs couleurs, le comportement des « shaders » de rendu, les propriétés de la caméra de la scène ainsi que la couleur d’arrière-plan de la scène sont tous paramétrés et modifiables. Il est possible d'ajuster la luminosité de l'image rendue à l'aide d'un éditeur de courbes, similaire à ceux offerts par des logiciels de manipulation photo comme "Photoshop". Les formes géométriques sont structurées dans plusieurs niveaux de calques qui forment une structure de scène, avec lesquels l'utilisateur est capable d'interagir afin de faire du traitement d'image et sur lesquels il est possible d'appliquer des filtres. À travers cette structure l'utilisateur peut aussi faire l’ajout et la sélection d’éléments. En plus de permettre à l'utilisateur de gérer de façon interactive les modes de vues dans l'application. Aussi, l'application comprend une interface graphique qui offre de la rétroaction informative visuelle à l’utilisateur et des contrôles interactifs pour influencer les états de l’application.



#2 Interactivité :

L'utilisateur peut interagir avec les lumières et les caméras grâce aux menus sur le côté. On peut en créer de nouvelle, et il est aussi possible de manipuler ceux qui existe déjà grâce à la souris. Il suffit simplement de les sélectionner dans l'environnement avec les souris et les déplacer en faisant traîner le curseur. Sur le menu de droite l'utilisateur peut voir les différentes couches de la scène, ce qui comprends les lumières, les plans ainsi que les formes géométriques existante. Il est possible de les sélectionner ou de les supprimer à travers ce menu, il suffit simplement de cliquer sur l'élément désiré. Pour contrôler la caméra, on peut utiliser la roue de la souris pour aller vers l'avant ou aller vers l'arrière. On peut aussi se déplacer dans le plan en faisant un « Click & Drag » dans l'un des deux environnements. On peut aussi utiliser le bouton « Shift » avec un « Drag » pour faire une rotation autour de la sélection.

Différents paramètres de l'engin de rendus peuvent être contrôlés, comme le nombre de bonds maximal, la contribution de l'illumination globale ainsi que leur nombre de 'samples', etc. Toutes les modifications sont affichées en temps réel à l'aide d'un buffer de rendu progressif inspiré de celui ajouté récemment dans Blender et V-Ray.

Les métadonnées des lumières comme la couleur, l'intensité, le rayon, l'atténuation ainsi que l'ombrage peuvent être contrôlé directement par l'utilisateur à travers le menu de gauche. Elles sont modifiables grâce aux curseurs de défilement étiqueté en conséquence.

#3 Technologie :

L'application a été conçue dans un environnement de travail créer sous Windows 10. Visual Studio 2015 ainsi que Xcode ont étaient utilisés pour créer l'application. Visual Studio 2015 a été utilisé spécifiquement pour améliorer la fusion du code entre les membres de l'équipe. La librairie OfxDatGui a été utilisée pour faire l'interface graphique de l'application puisqu'elle offre plus de possibilités que ce qui est offert par la librairie d'interface graphique offerte par openFrameWork. Les shaders GLSL ont été utilisés pour faire les filtres disponibles dans l'application ainsi que pour faire afficher les boîtes en mode survol (boite affichée lors du "mouse over"). Les addons **ofXDatGui** et **ofXDelauney** ont été utilisés pour le projet.

#4 Compilations :

Voici la marche à suivre pour être capable de compiler le projet :

1) Installer openFrameWork 9.8. Le site officiel d'openFrameWork offre un guide détaillé pour l'installation.

2) Télécharger le module ofxDatGui. Le module en question est inclus dans le document de remise dans le répertoire ofDatGui mais il peut aussi être téléchargé à partir du GitHub de la librairie ofxDatGui (premier résultat dans Google en cherchant «ofxDatGui»).

3) Copier le contenu du répertoire ofxDatGui dans le répertoire "addons" que l'on retrouve où openFrameWork a été installé (.....\of_v0.9.8_vs_release\addons.

4) Copier le contenu du répertoire ofxMathMesh-master dans le répertoire "addons" que l'on retrouve où openFrameWork a été installé (.....\of_v0.9.8_vs_release\addons.

5) Télécharger le module ofxDelaunay. Le module en question est inclus dans le document de remise dans le répertoire ofDelaunay, mais il peut être téléchargé sur le GitHub de la librairie ofxDelaunay (premier résultat dans Google en cherchant «ofxDelaunay»).

6) Copier le contenu du répertoire ofxDelaunay dans le répertoire “addons” que l'on retrouve où openFrameWork a été installé (.....\of_v0.9.8_vs_release\addons.

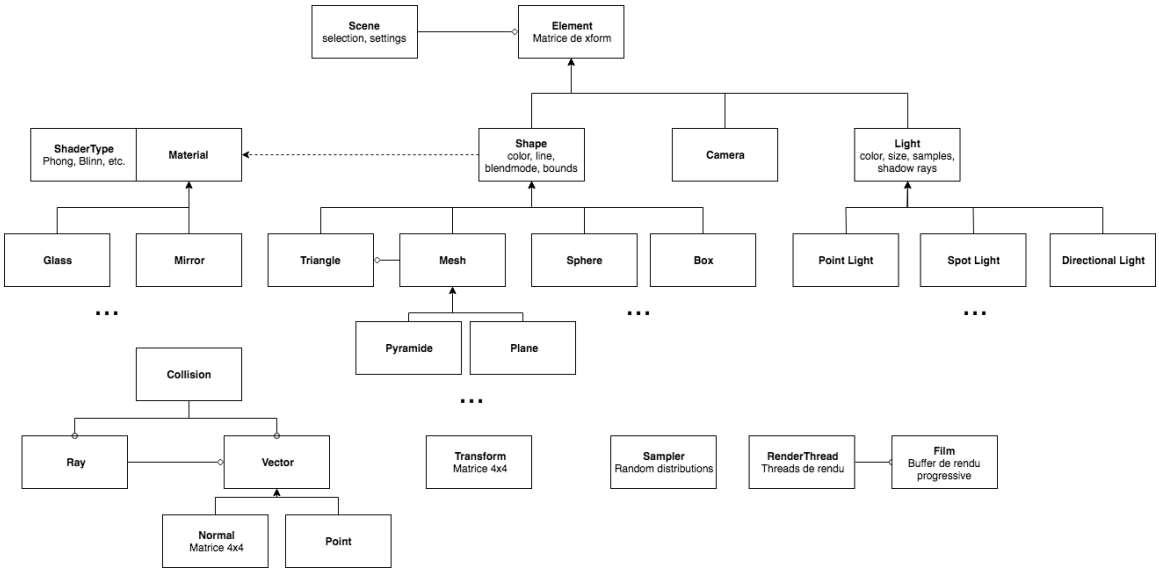
7) S'assurer que le répertoire “ofxbraitsch” se trouve bien dans le répertoire “\bin\data” du projet. S'il n'y est pas allé le chercher dans le répertoire ofxDatGui et copier-le dans le répertoire “\bin\data” du projet.

8) Le projet est prêt à être compilé. S'il ne marche pas, générer un nouveau projet avec le projet generator d'openFrameWork et importer les fichiers .cpp et .h du projet dans ce nouveau projet. S'assurer que les bon addons sont activés (voir #3).

#5 Architecture :

L'application a été construite au-dessus d’Openframeworks, de façon à être indépendante des librairies qui viennent avec l'environnement. Notamment, les classes de vecteur, matrices et de couleurs sont utilisées à travers l'application. Les classes ofNode et ofCamera sont héritées pour permettre la manipulation des objets de la scène. Cependant, elles sont toujours accédées à travers des classes héritées ou des typedefs, de façon à permettre leur remplacement et à faciliter l'extension de l'application vers d'autres plateformes ou libraires. Au niveau de la structure de classes, on retrouve encore une fois la superclasse Élément qui retient les propriétés partagées par tous les objets visibles, soit la position, la rotation, échelle. Tous les objets positionnables dans l'espace 3D héritent de la classe "élément", et implémentent la méthode virtuelle 'draw' et 'intersect' pour permettre l'affichage dans le viewport de "preview" ainsi que la sélection. La classe scène gère la sélection des objets ainsi que les propriétés globales de l'application. La classe “Shape” gère les attributs propres aux différentes formes, et contient un lien vers le

matériau de l'objet. En général, l'application fait énormément usage de l'héritage. Sa structure est expliquée par le diagramme suivant:



On y remarque notamment les liens d'héritage et le lien d'appartenance entre les différents objets.

#6 Fonctionnalités :

6.1 Point de vue

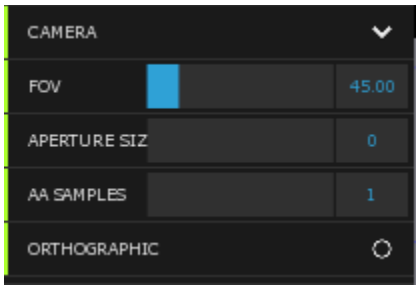
L'application implémente une caméra dont la position et la rotation peut être contrôlée de façon similaire aux applications 3D conventionnelles.

Il est possible de 'zoomer' avec le scroll de la souris ainsi que les flèches du clavier. On peut également déplacer la caméra avec la souris et le clavier, tant en termes de translation que de rotation, et autour d'une sélection.

Controls:

- Scroll-wheel pour “zoomer”
- Click & Drag for “panner”
- Shift-Click & Drag pour une rotation autour de la sélection

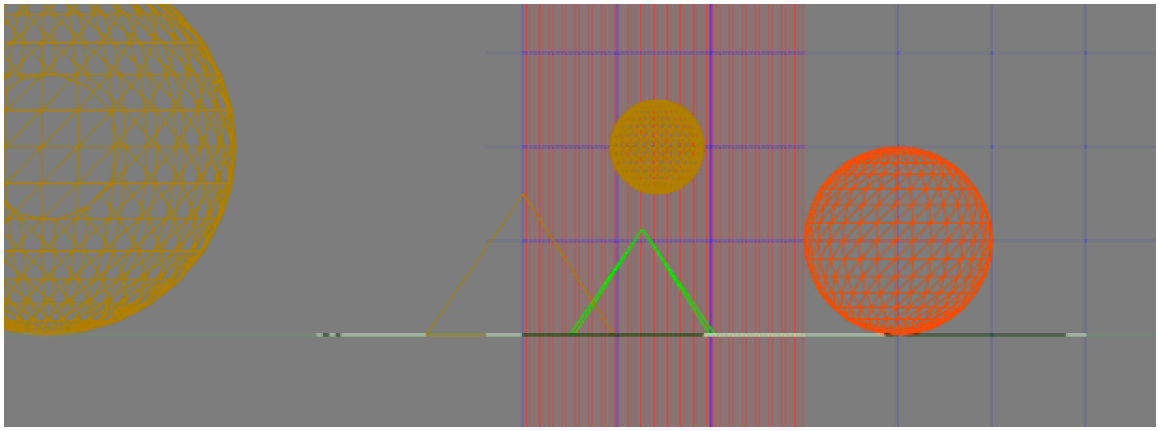
De plus, il est possible de sélectionner les éléments 3D avec un click de la souris et de déplacer ces derniers avec les flèches du clavier. Appuyer sur alt pour effectuer des plus petits déplacements.



La caméra supporte également d'autre paramètres comme la taille de l'ouverture et l'oversampling des rayons primaires, qui influencent l'aspect du rendu. Un mode de projection orthographique est également supporté pour le viewport.

6.2 Mode de projection

Les modes de projection en perspective et orthographique sont supportés pour les viewports. Il est possible to passer en mode orthographique en appuyant à travers le menu ou en appuyant sur 'u'. L'application effectue des calculs de perspective pour les calculs des rayons primaires de la caméra, de façon à équivaloir à la matrice de projection utilisée pour rendre le viewport. De plus, le FOV est paramétrisé et contrôlable à partir de l'interface.



6.3 Agencement

L'application offre deux vues principales:

- Le 'viewport' permettant l'affichage clair des objets et permettant la sélection de ceux-ci.

Ce dernier est affiché au bas de l'écran.

- Le 'renderview' permettant l'affichage du buffer de rendu et du progrès de l'algorithme de path tracing.

Le rendu progressif est supporté, et donc la mise à jour de cette vue se fait en continue. Une interface est affichée autour de ces deux vues offrant les différents contrôles de l'application.

6.4 Occlusion

Des bounding volumes en forme de sphère sont utilisés au niveau de maillages géométriques pour éviter les calculs superflus au niveau des calculs d'intersection des triangles. Lorsque l'intersection échoue avec la sphère englobant un ensemble de triangles, ces derniers sont 'cullés' et ignorés par le reste de la procédure.

7.1 Modèles d'illumination

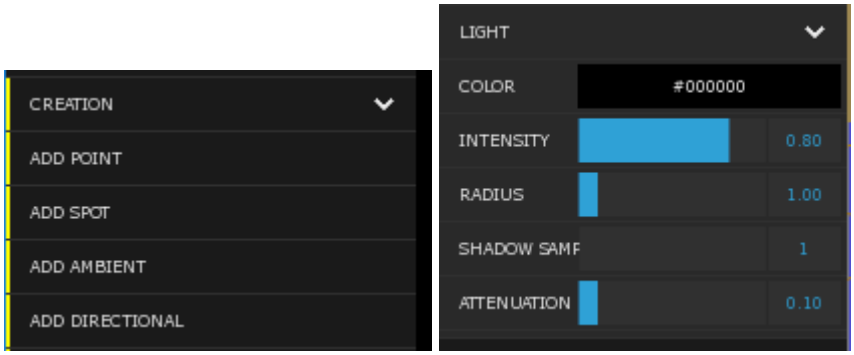
Les trois types de modèles l'illumination standards, soit lambert, phong, et blinn-phong sont implémentés et utilisés par différentes classes de matériaux. L'intensité du rehaut spéculaire peut être sélectionnée par un paramètre de 'specular glossiness'. Différents effets plus complexes sont ajoutés à ces modèles de basés et peuvent être combinés pour génerer des effets intéressants, comme par exemple un mélange de réflexion diffuse et spéculaire, et ainsi que de refraction. À ces modèles sont ajouté de l'illumination globales paramétrisée ('indirect' dans le panneau).

7.2 Matériaux

Un infinité de différents types de matériaux peuvent être composés en mélangeant les différents paramtères du panneau ‘Materiaux’. Plusières classes ont été implémentés pour des matériaux convientionnels comme l’effet mirroir, le verre, le plastique, etc. Plusieurs matériaux sont démontrés dans la scène chargée par défaut.

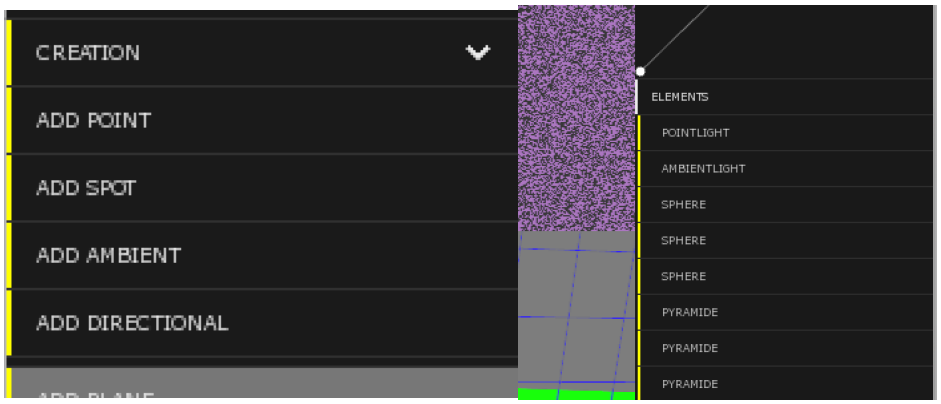
7.3 Types de lumière

Les quatre types de lumières paramétriques classiques sont offertes: Point Light, Directional Light, Spot Light, Ambient Light. Celles-ci peuvent être déplacées dans la scène et leur propriété de couleur modifies. Leur contribution est calculée par rapport à la position vis-à-vis la normale de la surface. Cette dernière est seulement ajoutée à l'équation de rendu s’il n'y a pas d'objet intermédiaire bloquant les rayons le lumière (shadow ray). De plus, les lumières supportent un contrôle de 'taille' qui permet le tracé d'ombre douces « raytracées ». Le nombre de samples utilisées pour ces calculs peut être contrôllées car ce type d’ombrage est notoirement lent à rendre.



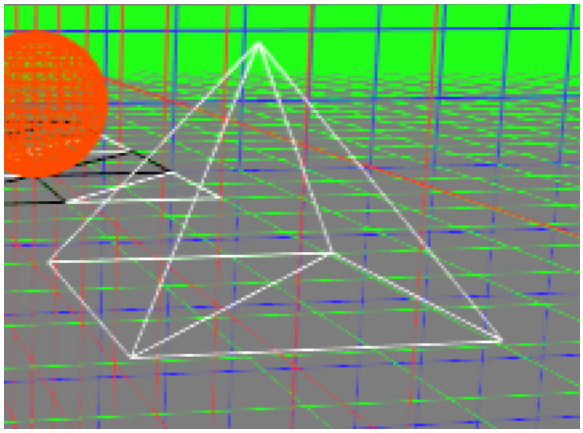
7.5 Lumières multiples

Il est possible d'ajouter une infinité de lumières à la scène et de contrôler leur position ainsi que leur couleur, leur intensité et leur atténuation à partir de l'interface. La taille de l’ombre douche ainsi que le nombre de samples peut également être choisi. (Note : avec 1 seul sample les ombres douces ne sont pas activées.



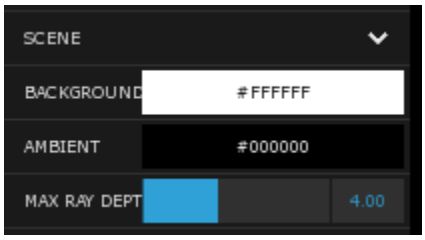
8.1 Intersection

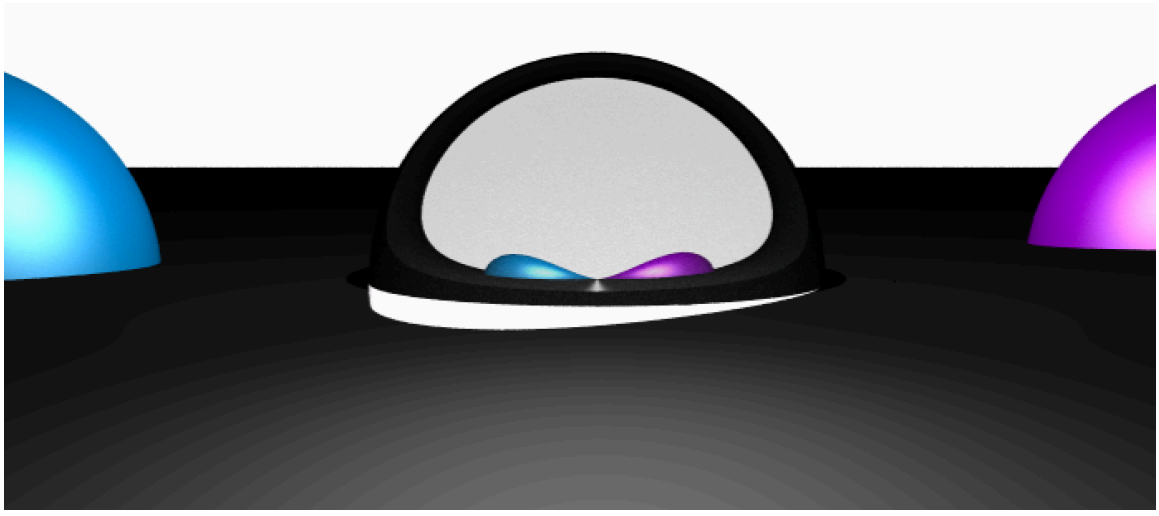
L'intersection entre les rayons et de nombreuses formes paramétriques comme les sphères, les plans, les disques, les boîtes AABB et les triangles sont supportées et ont été implémenter à la main. Ces dernières sont utilisées pour rendre et également pour calculer la sélection et le culling des bounding volumes. Des maillages géométriques peuvent être formés à partir de triangles pour former des formes plus complexes.



8.2 Réflexion

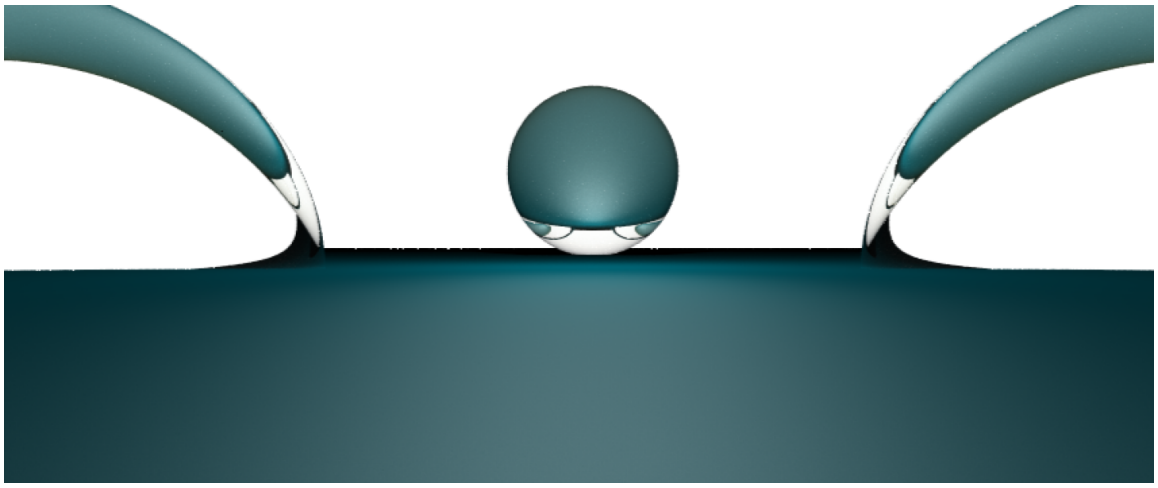
La réflexion par raytracing est implémentée, et l'intensité de cet effet peut être contrôllé et mélangé aux autres effets de shader. Puisque la réflexion est un algorithme récursif, la profondeur maximale est paramètrée et peut être chosie dans le menu 'Scène'.





8.3 Réfraction

La réfraction est supportée naturellement par l'algorithme de ray tracing et offerte à travers le paramètre 'réfraction' des matériaux. L'IOR est également supporté et permet de contrôler l'angle de réfraction; la contribution de celle-ci est modulée par l'équation de fresnel, ce qui permet plus de réalisme lorsque la réflexion et la réfraction sont combinés.

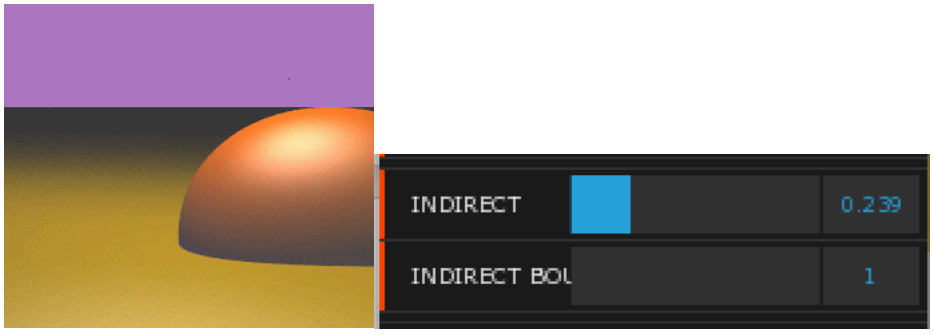


8.4 Ombrage

Les ombres simples par shadow sont implémentées pour toutes les types de lumières primitives appropriées. En plus, des ombres douces complexes peuvent être formées avec un paramètre de ‘radius’ des lumières, ce qui permet de le calcul d’ombres raytracés.

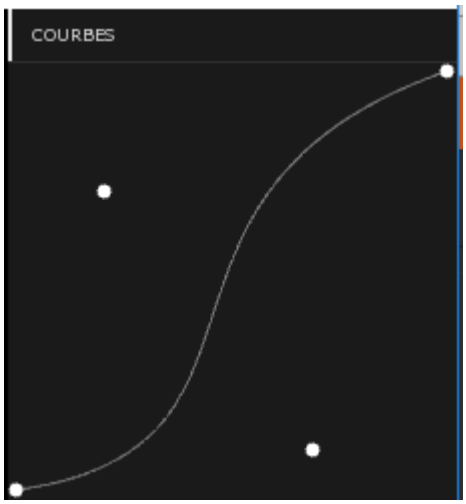
8.5 Illumination globale

Illumination indirecte est supportée directement, par raytracing Monte Carlo. Aux points d'intersection, des rayons secondaires sont 'samplés' dans l'hémisphère de la normale et tracés récursivement pour évaluer la lumière incidente de façon réaliste.



9.1 Courbe cubique

L'application offre des contrôles d'ajustement de couleur, qui sont paramétrisés par l'utilisateur via des courbes de contrôles. Celles-ci utilisent une courbe cubique pour interpoler entre des points de contrôles spécifiés par l'usager, et ainsi contrôler l'ajustement en plus d'afficher l'outil dans l'interface.

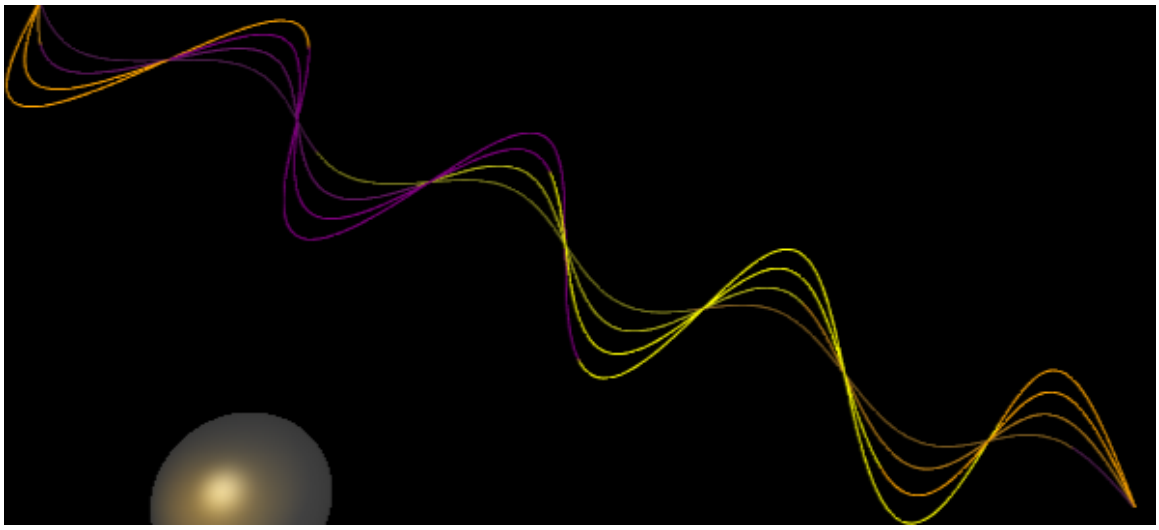


9.2 Courbe paramétrique

L'outil permet également d'utiliser le contrôle par courbe de Bézier, ce qui est souvent plus pragmatique. Différents points de contrôles peuvent être créés et manipulés.

9.3 Surface paramétrique

Ce critère est implémenté dans l'application offre la possibilité d'afficher une surface paramétrique de Bézier qui simule l'apparence du courbe de sinus. On l'affiche en appuyant sur le bouton « Surface paramétrique » dans le menu du UI.

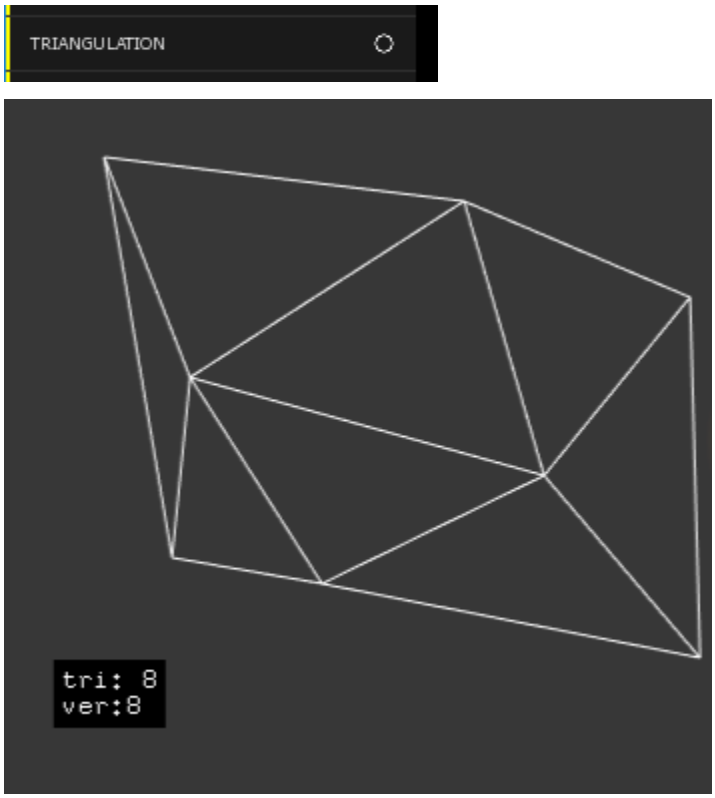


9.4 Shader de tesselation

La surface paramétrique utilise de la tesselation.

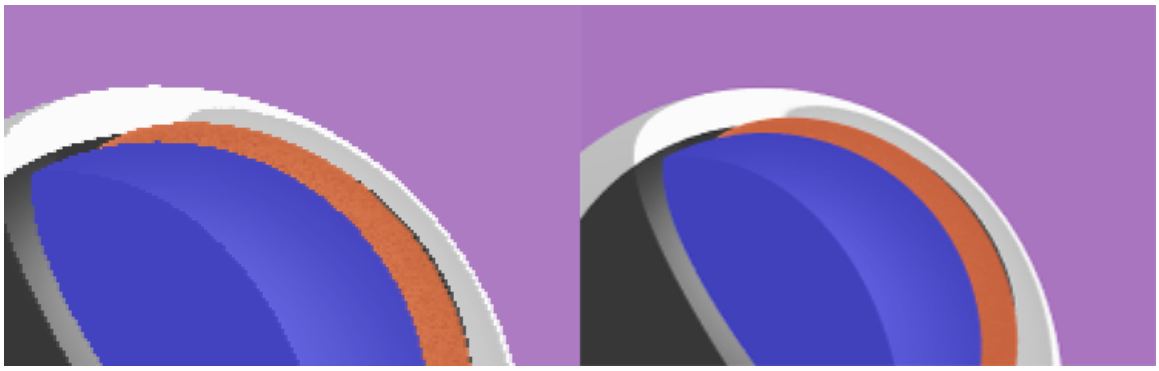
9.5 Triangulation

En appuyant sur le « Toggle » Triangulation dans le dossier création du GUI, il est possible de créer des points aux endroits où on clique afin de créer des triangles lorsque trois points ou plus sont placés. L'algorithme utilisé est l'algorithme de triangulation de Delaunay.



10.1 Effet en pleine fenêtre

Des contrôles de correction couleur (brightness) sont offerts à l'utilisateur ce qui permet d'apporter des ajustements finaux à l'aspect du rendu. L'anticrénelage est supporté par oversampling des pixels lors du ray casting, ce qui améliore grandement la qualité visuelle du rendu. Finalement, un effet de flou de lentille est offert, et implémenté en traçant des rayons primaires à partir d'un disque plutôt que d'un point au niveau de l'origine de la caméra.

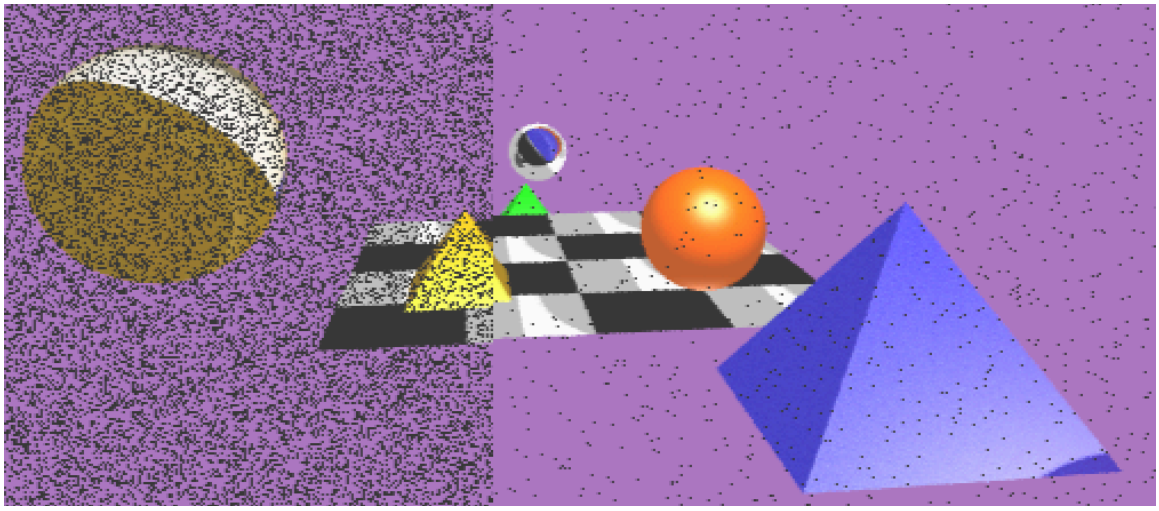


10.3 BRDF

Les capacités d'illumination globale utilisent une équation de BRDF pour pondérer l'irradiance selon l'angle incident lors de la sommation de l'algorithme de Monte Carlo. Un modèle simple est implémenté mais le même engin peut facilement supporter des équations plus complexes par simple overloading de la fonction 'BRDF' de la classe Shader.

10.4 Rendu en différé

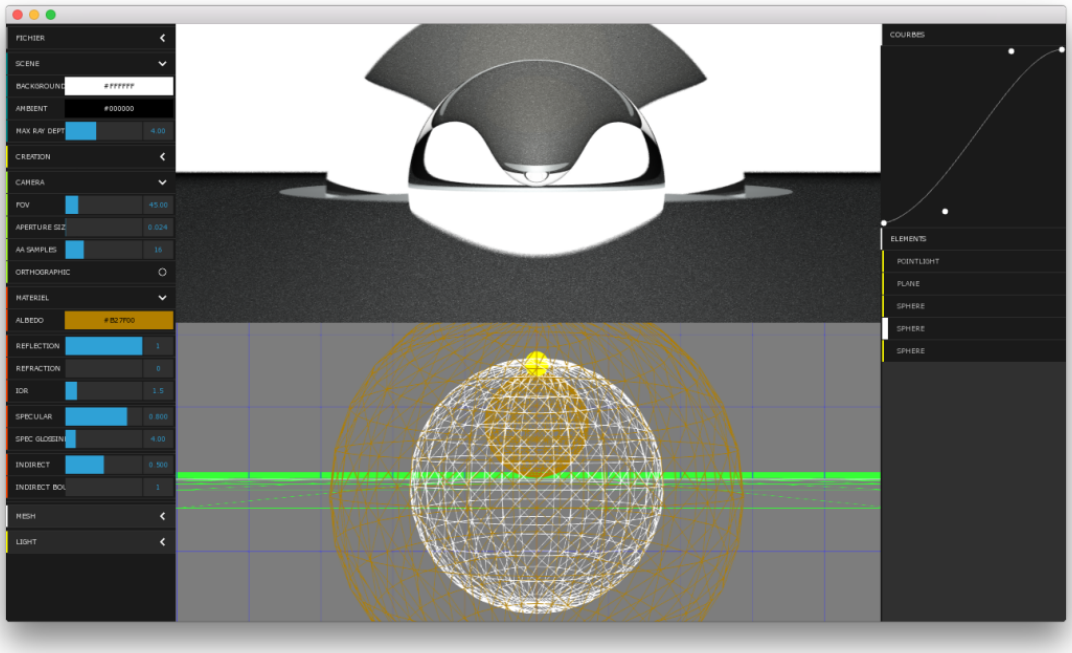
L'application implémente du rendu progressif, qui permet d'offre du 'feedback' rapide à l'utilisateur lors de rendus qui sont parfois long. Les pixels sont accumulés dans un buffer qui maintient une 'moving average' des samples jusqu'à présent, et ainsi il est possible de voir de voir un résultat préliminaire de la scène même si seulement quelques samples ont été calculés pour un pixel. L'écran est automatiquement rafraîchi lorsque la scène est modifiée.

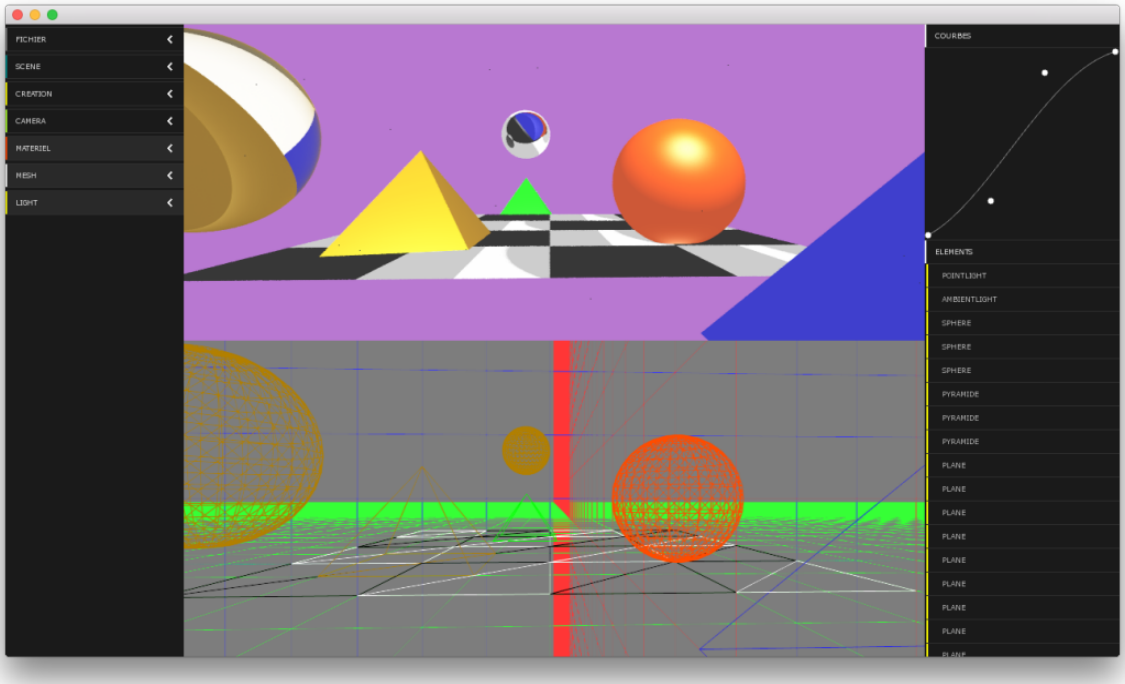


10.5 Style libre : Multithreading

L'application implémente du multithreading pour accélérer le rendu. Différents threads écrivent de façon concurrente sur le buffer de rendu. L'intégrité est maintenue par la commutativité de l'ajout de 'samples'; peu importe l'ordre dans lesquels ils sont ajoutés, leur moyenne sera la même.

Voici une série d’images générées à partir de notre application :



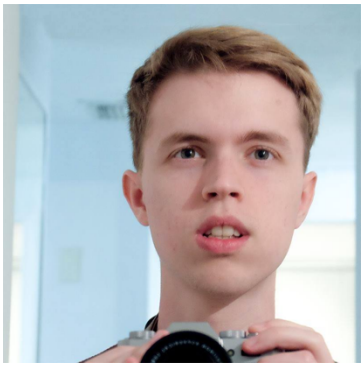


#7. Ressources

Pour les fonctions refraction() et fresnel() de la classe vendor.cpp, nous nous inspirer d'un article sur le site suivant : www.scratchapixel.com.

#8. Présentation

Vincent Côté Giroux est diplômé du Cegep François-Xavier Garneau depuis décembre 2015, à la recherche d'un emploi de technicien en informatique. Mes réalisations rémunérées ainsi que de stagiaire m'ont permis de démontrer mes capacités de développeur Web et mobile avec l'approche agile, et de soutien à la clientèle dans un Centre de services technologiques.



Yohan Poirier-Ginter a complété un DEC en Animation 3D au Cégep de Limoilou. Il poursuit dorénavant divers intérêts dont l'informatique à travers un BAC à l'Université Laval.

Michaël Dodier est diplômé du Cégep de Thetford depuis juin 2016. Il a ensuite débuté son Bac. En informatique à l'université Laval. Par ailleurs, il travaille actuellement comme programmeur à temps partiel pour la compagnie Référence Systèmes en plus de faire ses cours. Il se passionne plus particulièrement pour le développement d'applications web.