INFOGRAPHIE
IFT-3100

Document de design

Projet de session - livrable 2

Préparé par :
Damien Arroyo
Marc-André Tremblay
Marcel Samson Morasse
Rémy Tremblay

30 Avril 2018

1. Sommaire

Ce projet met en pratique les 10 modules du cours d'infographie IFT-3100. Pour le deuxième livrable, les 5 derniers modules, qui sont les camera, l'illumination, le lancer de rayon, la topologie et les techniques de rendu, ont été ajoutés. En utilisant les connaissances acquise grâce à ces modules, les étudiants ont eu comme mandat de développer de nouvelles fonctionnalités pour l'application de traitement d'image à la photoshop du livrable 1.

Au total, 50 critères fonctionnels sont proposés. Ceux-ci sont divisé en 10 catégories. Il est proposé de faire l'implémentation d'un minimum de 3 critères par catégorie afin que l'application paraisse pleinement fonctionnel et réponde aux besoins des utilisateurs.

Pour ce projet les choix de technologie, le design logiciel et la thématique du projet est à la discrétion des étudiant. Ainsi, les étudiant on choisi de faire une application ayant une interface de type photoshop, c'est-à-dire, simple et concise mais tout en donnant la possibilité aux utilisateurs qui cherche à faire du traitement de scène plus poussé de le faire. La librairie OpenFrameworks est le pilier de plusieurs des fonctions lié aux traitement d'image et de scène ainsi que son addon ofxDatGui pour l'interface.

Une architecture utilisant le patron de conception MVC (model, view, controller) est utilisée afin de séparer la logique du domaine aux éléments de l'interface. Cette approche favorise la clarté du code, la responsabilité unique, un faible couplage et une forte cohésion.

Le présent document présente les principales composantes du processus de développement de l'application en question et cela en passant par la justification des choix de conception ainsi que la description complète de ceux-ci.

2. Interactivité

Afin d'interagir avec notre application, nous avons fait plusieurs menus permettant de contrôler plusieurs parties différentes de l'application.

2.1 Aperçu d'une scène avec quelques objets et leur panneau de contrôle.

Le "control panel" permet d'importer des images et d'exporter le contenu du canvas ainsi que d'ajouter des primitives 2D et 3D au canvas et de les retirer en les sélectionnant dans la liste d'objets ajoutés à l'écran.

Le menu positionné sous le "control panel" permet de sélectionner un ou plusieurs primitives 2D afin de modifier leur couleur simultanément. Il est aussi possible de retirer les primitives sélectionnées en cliquant sur "Remove Selected".

Finalement, chaque primitive créée affiche un menu de primitive permettant de modifier tous les attributs de cette primitive. Par exemple, un cercle nous permet de modifier sa position en X et Y, changer son rayon, la largeur de son trait extérieur, sa couleur primaire et secondaire et, finalement, de choisir si on veut que ce cercle soit rempli ou non. Ces attributs changent dépendant des primitives.

3. Technologie

L’application a été développée dans l'environnement de développement Visual Studio 2015. En effet, tous les membres de l’équipe utilisent Window, alors cet outil paraissait le plus approprié et il est également proposé sur le site officiel d’OpenFrameWorks.

L’application a été programmée en C++ avec la librairie OpenframeWorks. Cette librairie permet de communiquer plus facilement avec les fonctionnalités OpenGL de l’ordinateur et offre plusieurs outils qui répondent au besoin de l’application.

Quelques “addons” d’OpenframeWorks ont été utilisés, soit ofxDatGui pour créer l’interface. Cet ‘’addon’’ n’en est pas un officiel de Openframeworks. Ainsi, afin d’y avoir accès, il faut cloner un répertoire dans le dossier ‘’addons’’ d’openframeworks. Cette librairie offre beaucoup plus de fonctionnalitées que ofxGui (version officielle de openframeworks) et a une meilleure documentation. Grâce à celle-ci, il est possible de faire des menus déroulants et d’avoir accès à des sélecteurs de couleur dynamiques pour ne nommer que ceux-ci. Également, ofxAssimpModelLoader à été utilisé pour permettre de travailler avec des modèles 3D.

L’importation et l’exportation d’images utilise la librairie “Common item dialog” de Microsoft afin de faire apparaître l’Explorateur Windows pour choisir un fichier sur l’ordinateur. Ce programme ne peut donc être utilisé qu’avec un système d’exploitation Windows Vista ou un autre système d’exploitation Windows plus récent.

Pour les surfaces de Bézier, la librairie ofxBezierSurface été utilisée. Elle permet de prendre les points de contrôles créés par l’utilisateur et de les afficher sous forme de surface.

Pour la triangulation de Delaunay, la librairie ofxDelaunay été utilisée. C’est celle-ci qui gère l’ajouts de points et qui affiche le maillage triangulaire.

L’addon ofxPostProcessing a été utilisé pour appliquer différents effets visuels sur toute la fenêtre d’affichage. Cette librairie utilise des techniques de rendu sur les images et les modèles 3D.

4. Compilation

Prendre note que cette application n'est pas utilisable sur Mac et linux. Elle Fonctionne uniquement sur Windows Vista et plus récent.

L'exécutable déjà compilé est TP2/bin/TP2_debug.exe. Le programme peut donc être exécuté en double-cliquant sur ce fichier.

Voici les étapes pour compiler le projet avec Visual Studio 2015 :

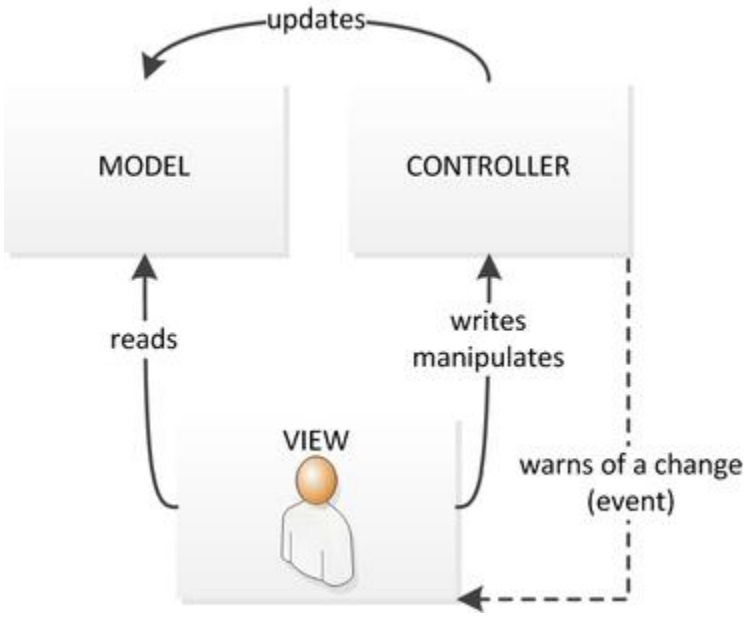
1. Avoir les librairies d'openframeWorks sur votre ordinateur.
2. Faire un clone du projet dans ...\of_v0.9.8_vs_release\apps\myApps, ou faire un pull si vous avez déjà le repo sur votre ordinateur à cet endroit.
3. Modifier le chemin d'importation du of_v0.9.8_vs_release. Pour ce faire, ouvrir visual studio 2015, faites un clique droit sur le projet TP2 dans le Solution Explorer, puis cliquer sur "edit".
4. Ensuite, mettre le projet TP2 comme projet par defaut en faisant clique droit puis "Set as startUp project".
5. Aller à la page suivante: https://github.com/braitsch/ofxDatGui. Copiez l'url du repo et le cloner dans of_v0.9.8_vs_release\addons. Ensuite, ouvrir le dossier ofxDatGui nouvellement créé dans addons et copier le dossier ofxbraitsch. Coller ce dossier dans of_v0.9.8_vs_release\apps\myApps\infographie-TP2\TP2\bin\data. L'option d'ajout de ce addon sera dans la section community addons.
6. Aller à la page suivante: https://github.com/charlesveasey/ofxBezierSurface. Copiez l'url du repo et le cloner dans of_v0.9.8_vs_release\addons. Cocher l'option d'ajout de ce addon dans la section community addons en cliquant droit sur le projet et en allant dans openFrameworks addons.
7. Aller à la page suivante: https://github.com/neilmendoza/ofxPostProcessing/tree/stable. Télécharger le zip du repo et copié le dossier dans of_v0.9.8_vs_release\addons (Cloner ne fonctionnera pas). Cocher l'option d'ajout de ce addon dans la section community addons en cliquant droit sur le projet et en allant dans openFrameworks addons.
8. Aller à la page suivante: https://github.com/obviousjim/ofxDelaunay. Copiez l'url du repo et le cloner dans of_v0.9.8_vs_release\addons. Ensuite, ouvrir le fichier ofxDelaunay.cpp dans le dossier de cet addon et supprimer la ligne 118 (ofDrawBitmapStringHighlight("tri: " + ofToString(ntri) + "\nver:" + ofToString(vertices.size()), 30, 300);). L'option d'ajout de ce addon sera dans la section community addons.

9. Importer les addons de openframeworks. Pour cela, vous devez faire clique droit sur TP2, puis selectionner "openFrameworks Addons". Cocher les addons : ofxAssimpModelLoader, ofxGui, ofxXmlSettings, ofxBezierSurface, ofxDatGui, ofxDelaunay et ofxPostProcessing-stable.
10. Modifier le chemin des dépendances additionnelles. Faire un clique droit sur le projet dans le Solution Explorer et aller dans Properties -> Linker -> Input. Cliquer sur le chemin de Additional Dependencies, puis <Edit...>. Effacer les premiers caractères au début du chemin "%3b" et s'assurer que le chemin n'est pas écrit en double. Appliquer les changements.
11. Cliquer sur le bouton “Local Windows Debugger” en haut au centre de la barre de menu de Visual Studio 2015.

5. Architecture

L'application respecte une architecture de type Modèle, Vue, Contrôleur. En effet, le patron de conception MVC est le pilier de la structure du projet. Grâce à cette approche, il est possible de séparer la logique du domaine aux éléments de l'interface. Cette approche favorise également la clarté du code, la responsabilité unique, un faible couplage et une forte cohésion. L'image suivante démontre les dépendances et liens entre les composantes du projet selon le model MVC:

5.1 Démonstration du flux de données dans un modèle respectant le patron MVC.

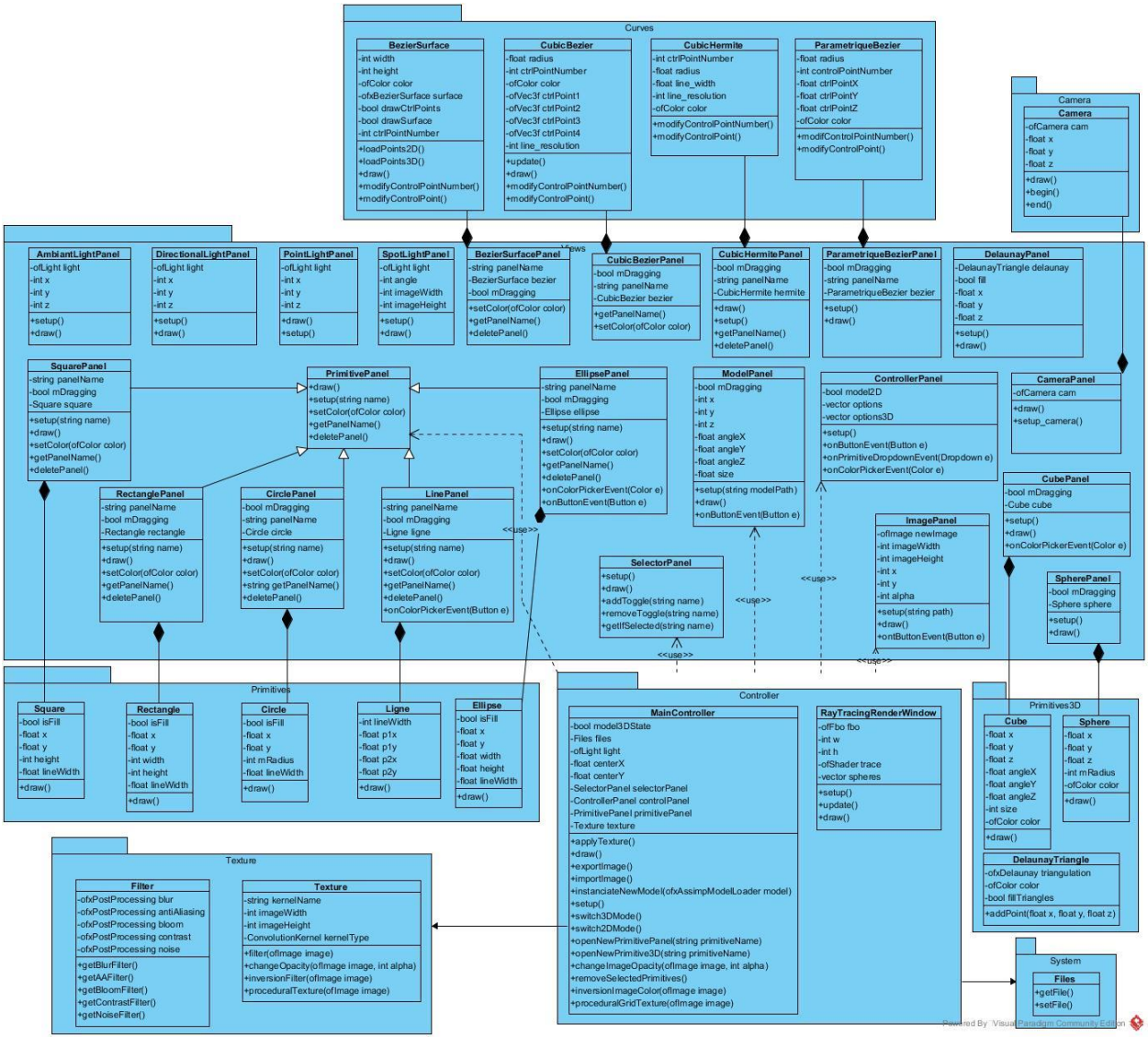


Dans ce projet, les modèles représentent les primitives 2D et 3D et les images et modèles importés que l'application peut afficher. Ceux-ci ne possèdent que 2 méthodes : leur constructeur et Draw(). Hormis ces 2 méthodes, les modèles ne possèdent aucune logique, ils ne servent qu'à conserver l'état des primitives selon différents paramètres. Ces paramètres sont modifiés par les contrôleurs uniquement. Ceux-ci s'occupent de la logique du domaine. Dans le cas de cette application il y a 2 contrôleurs. En effet, mainController s'occupe de l'état global de l'application. Il permet, en autre, de supprimer les panneaux de contrôle de primitives, de passer du mode 2D à 3D, importer des images, etc. Le deuxième contrôleur n'a pas été implémenté par l'équipe de projet. Effectivement, ce contrôleur est fourni par l'addon ofxDatGui. Celui-ci permet de lier les modèles aux vues et de modifier les modèles en temps réel dans l'application. Les vues, quant à elles, s'occupent de tout ce qui est affichage graphique et envoient les événements aux contrôleurs. Puisque les vues sont liées à leur modèle

correspondant, lorsqu'un des modèles est modifié par le contrôleur, la vue se modifie en conséquence.

Enfin de mieux comprendre l'architecture, voici le diagramme de classe de l'application:

5.2 Diagramme de classe de l'application.



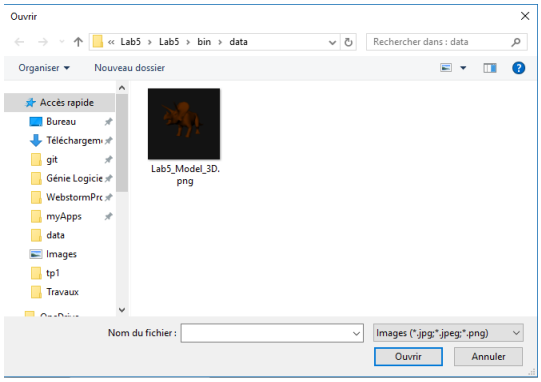
6. Fonctionnalités

6.1 Image

6.1.1 Importation d'image

Il est possible d'importer des fichiers images .jpg, .jpeg et .png, ainsi que des modèles 3D .obj et de les afficher à l'écran. En appuyant sur le bouton "Import" situé dans le menu de gauche, une fenêtre de dialogue Windows s'ouvre afin de choisir le fichier à ouvrir. Cette fenêtre permet aussi de filtrer les fichiers qu'il est possible d'importer selon leur format (.jpg, .jpeg, .png, .obj) en cliquant sur le menu déroulant en bas à gauche.

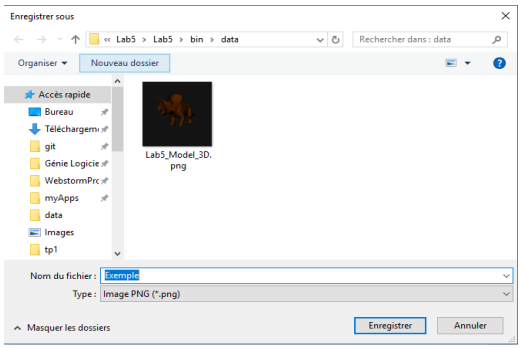
6.1.1.1 Fenêtre d'importation de fichiers.



6.1.2 Exportation d'image

Il est possible d'exporter des fichiers images .png. En appuyant sur le bouton "Export" situé dans le menu de gauche, une fenêtre de dialogue Windows s'ouvre afin de choisir l'emplacement du fichier à enregistrer et du nom à donner au fichier. L'image exportée correspond à toute la zone d'affichage du programme, en dehors du menu.

6.1.2.1 Fenêtre d'exportation de fichiers.



6.1.3 Sélecteur de couleur

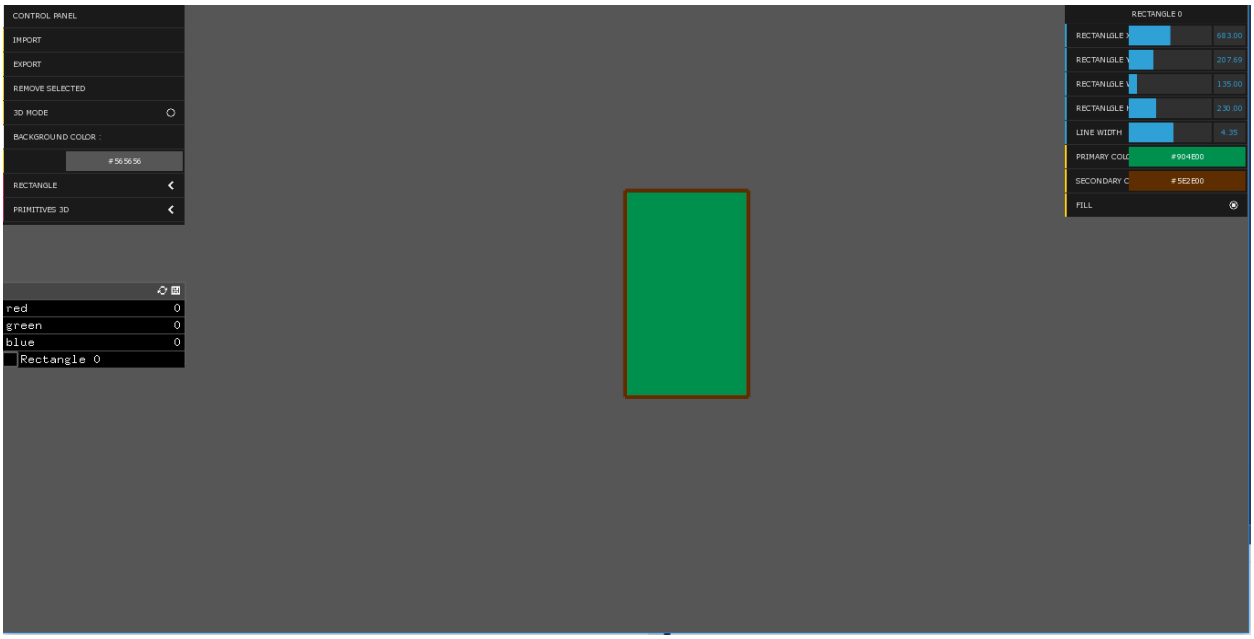
Il est possible de sélectionner une couleur parmi un ensemble de couleurs et de l’assigner à un élément visuel. La couleur peut être appliqué à l'arrière-plan ainsi qu’aux primitives. Le sélecteur de couleur est disponible dans chaque fenêtre pour être appliqué à l’élément désiré.

6. 2 Dessin vectoriel

6.2.1 Outil de dessin

L’application permet de modifier de manière interactive la valeur des outils de dessin vectoriels tel que l’épaisseur des lignes de contour, la couleur des lignes de contour, la couleur des zones de remplissage et la couleur d’arrière-plan de la scène. Ces fonctionnalités sont accessibles grâce au panneau de contrôle qui est instancié lors de la création d’une primitive. Chaque panneau est différent selon la primitive instanciée. En voici un exemple:

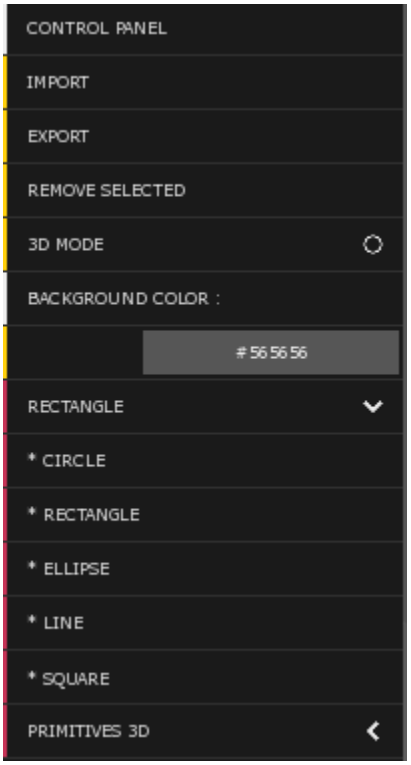
6.2.1.1 Panneau de controle des propriétés de la primitive rectangle.



6.2.2 Primitives vectoriel

Il est possible de créer de manière interactive des instances pour ces 5 types de primitives vectorielles : ligne, carré, rectangle, cercle et ellipse. Ces primitives vectorielles sont sélectionnables à partir du menu déroulant ‘’Primitive 2D’’ dans le panneau de control principal. Voici à quoi ressemble ce menu déroulant et ces options.

6.2.2.1 Menu déroulant des primitives 2D dans le panneau de contrôle principal.



Lorsqu’un primitive est sélectionnée, elle est instanciée au milieu de l’écran avec des dimensions et couleurs prédéfinies. Également, les primitives sont vides lors de leur instanciation. Pour les remplir de leur couleur primaire, il faut cocher l’onglet ‘’Fill’’ dans le panneau de control correspondant a la primitive que l’on désir remplir.

6.2.3 Interface

Des éléments d’interface graphique offre de la rétroaction informative visuelle à l’utilisateur et des contrôles interactifs pour influencer les états de l’application. En effet, le panneau de contrôle principal permet de passer du mode 2D au mode 3D et également d’offrir un contrôle sur la couleur d’arrière-plan de l'application. L’ensemble des panneaux de contrôle disponibles dans l’application permettent à l’utilisateur d’avoir

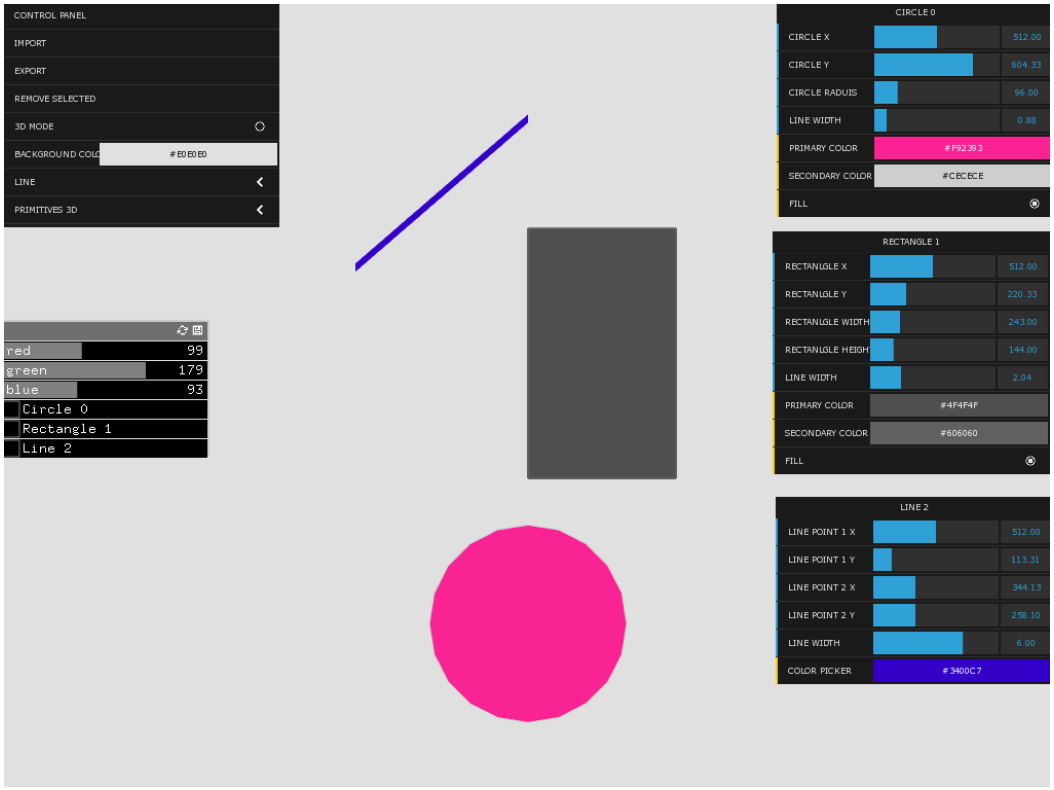
une rétroaction sur ce qu'il fait dans l'application. Entre autre, le panneau qui gère les éléments de la scène permet de savoir combien d'éléments se trouvent dans la scène courante et se modifie dynamiquement selon l'ajout ou la suppression de primitives. De plus, les panneaux de contrôle de primitives permettent de modifier en temps réel leur primitive grâce aux sliders qu'il fournissent. Il est donc possible, entre autre, de modifier la valeur du slider de la position en X de la primitive et de voir la primitive se déplacer en même temps dans la scène selon la nouvelle position apportée par la modification du slider.

6.3 Transformation

6.3.1 Graphe de scène

Il est possible d'ajouter, supprimer et sélectionner un ou plusieurs objets à l'écran afin de changer une ou plusieurs de leurs attributs, soit un à un ou plusieurs simultanéments en les sélectionnant dans le menu où le nom et numéro de l'objet apparaissent lors de sa création.

6.3.1.1 Aperçu d'une scène avec quelques objets.



6.3.2 Sélection multiple

Il est possible de sélectionner plusieurs objets créés et de les modifier simultanément. Cependant, certains aspects des objets ne peuvent pas être modifiés simultanément tel que leur position et leur grosseur. La couleur des objets sélectionnés peut être modifiée simultanément ou un à la fois.

6.3.2.1 Sélection et changement de couleur de trois éléments simultanément.

6.3.3 Transformation interactives

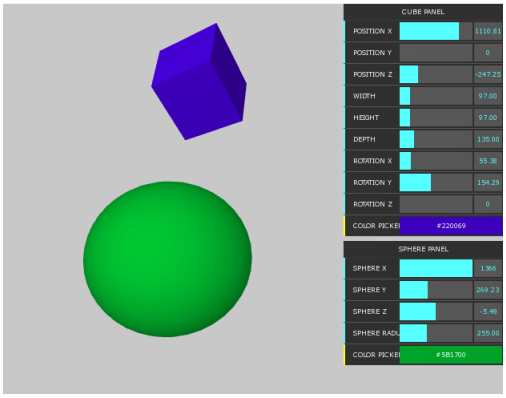
Il est possible de modifier de façon interactive la translation, rotation et proportion de plusieurs éléments visuels créés et présents sur le canvas. Certains éléments telles les primitives 2D ne supportent pas la rotation. Les autres transformations sont cependant possibles sur toutes les primitives 2D ou 3D et sont accessibles via l'interface graphique permettant de voir les changements en même temps qu'ils sont fait.

6.4 Géométrie

6.4.1 Primitives géométriques

Il est possible de dessiner 2 types de primitives géométriques 3D, soit une boîte et une sphère, à partir d'un algorithme qui n'utilise aucune donnée externe au programme. Ces primitives sont créées à l'aide de fonctionnalités d'OpenframeWorks. La couleur, la position en X, Y et en Z, ainsi que le rayon de la sphère peuvent être modifiés. La couleur, la position en X, Y et en Z, l'angle en X, Y et en Z, ainsi que la largeur, la hauteur et la profondeur peuvent être modifiés pour le cube.

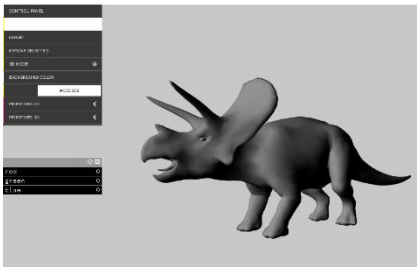
6.4.1.1 Exemple de primitives géométriques.



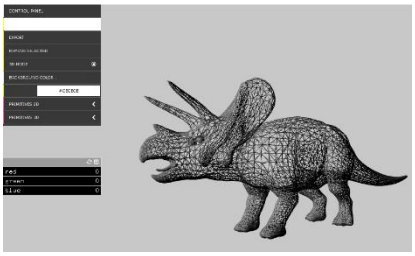
6.4.2 Modèle 3D

Il est possible de dessiner 3 types de modèles 3D importés à partir de fichiers externes .obj ou .dae. Ces types correspondent aux maillages triangulaires Fill, Wireframe et Points. Le premier représente le modèle avec toutes ses faces, le deuxièmes sous forme de sommets reliés par des arêtes et le troisième représente seulement les sommets du maillage. Il est possible de changer le type du modèle en cours d'affichage grâce aux boutons situés dans le panneau de contrôle de l'objet.

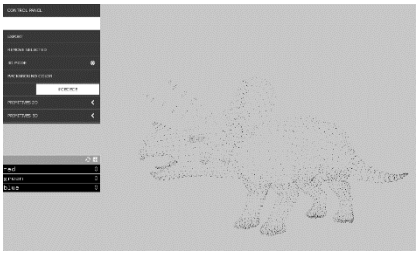
6.4.2.1 Modèle 3D de triceratops importé et représenté avec le type Fill.



6.4.2.2 Modèle 3D de triceratops importé et représenté avec le type WireFrame.



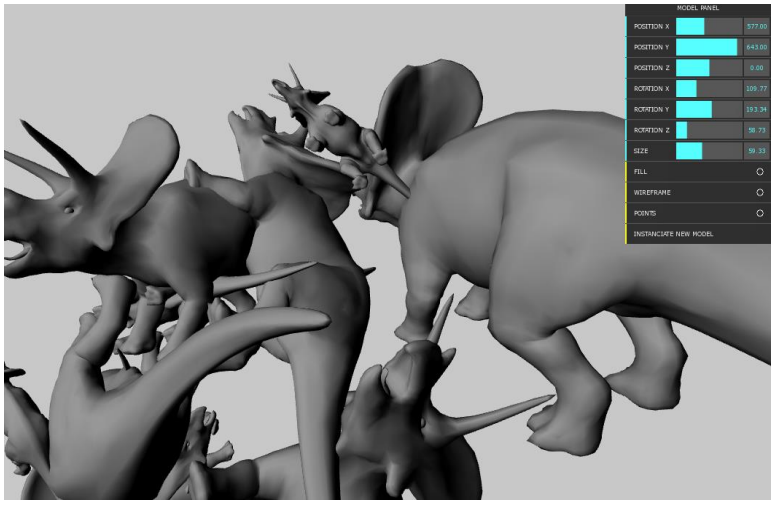
6.4.2.3 Modèle 3D de triceratops importé et représenté avec le type Points.



6.4.3 Instanciation

Il est possible de dessiner de multiples instances d'un modèle qui ont des transformations différentes avec une seule commande d'affichage. En appuyant sur le bouton "Instanciate New Model" dans le panneau de contrôle d'un objet, une nouvelle instance du modèle va apparaître, mais à des positions aléatoires en X, Y et Z, avec une proportion entre 1% et 150% du modèle original et dans des angles aléatoires en X, Y et Z. Il est possible d'appuyer sur le bouton autant de fois que voulu pour avoir plusieurs instances avec des transformations différentes.

6.4.3.1 Instanciation de plusieurs triceratops avec des transformations différentes

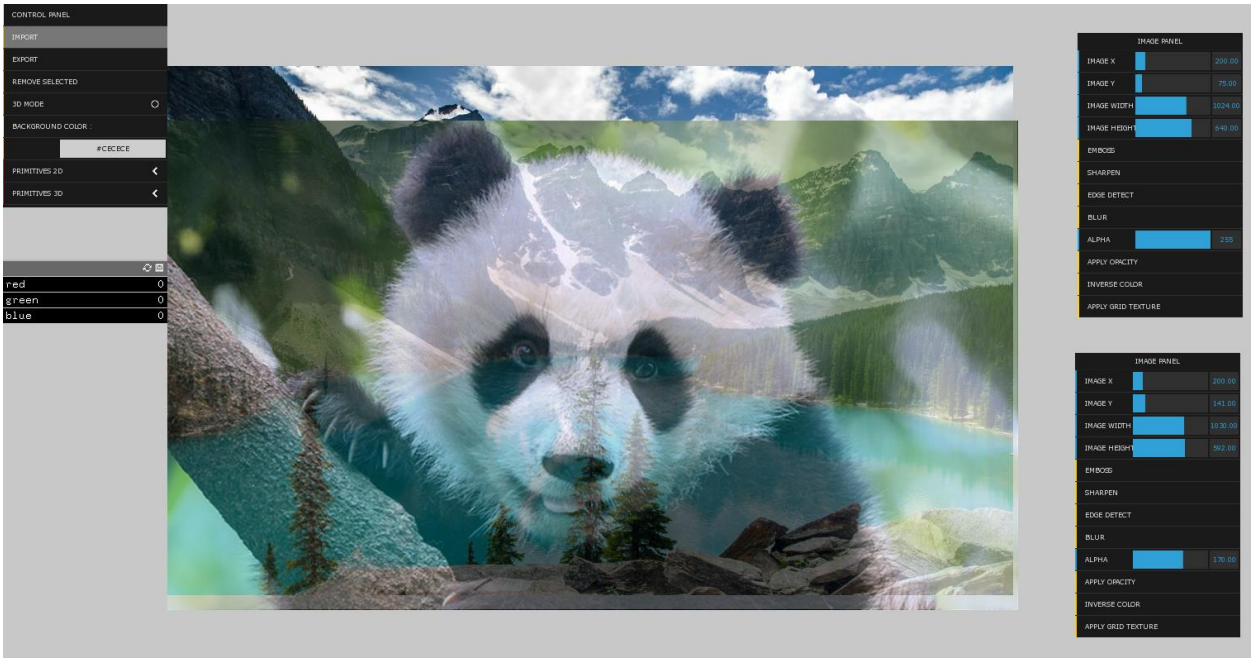


6.5 Texture

6.5.1 Composition

Lorsqu'une image est importée à l'aide du bouton "Import", un panneau de contrôle apparaît dans le haut à droite de la fenêtre et permet de modifier certains attributs de l'image. Il est donc possible de changer l'opacité de chaque image pour permettre d'en superposer plusieurs. La transparence de l'image est choisi à l'aide d'un "slider" et doit être confirmée avec le bouton "Apply opacity". Dans l'image qui suit, le paysage en dessous a une opacité de 100%, alors que l'image de panda au dessus a une opacité de 70%.

6.5.1.1 Image de panda avec une opacité réduite par dessus une image de paysage



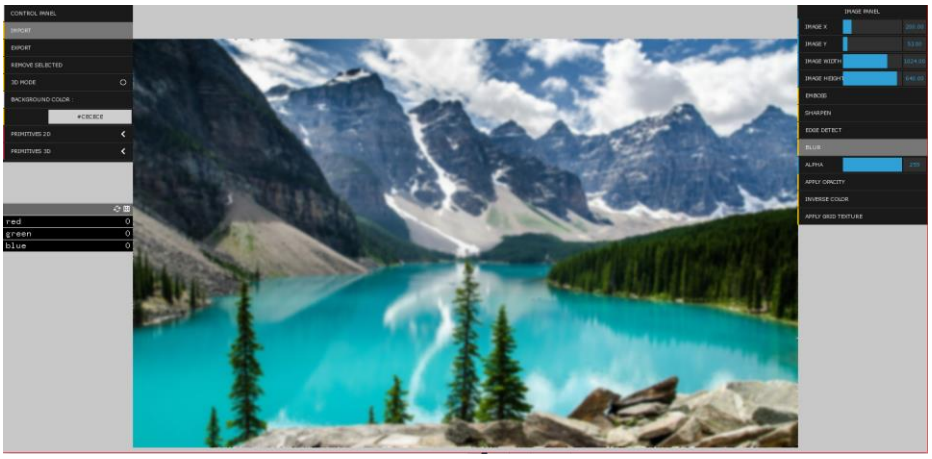
6.5.2 Traitement

À l'aide du même panneau de contrôle que la composition, il est possible d'appliquer plusieurs types de traitement d'image tel que le traitement flou, aiguisé, inversion de couleur, détection de bordure et bosselé. Un traitement s'applique sur l'image à chaque fois que l'on appuie sur le bouton de la texture voulue. On peut appliquer plusieurs textures différentes sur chaque image.

6.5.2.1 Filtre bosselé sur une image de paysage.

6.5.2.2 Filtre d’inversion des couleurs sur une image de paysage.

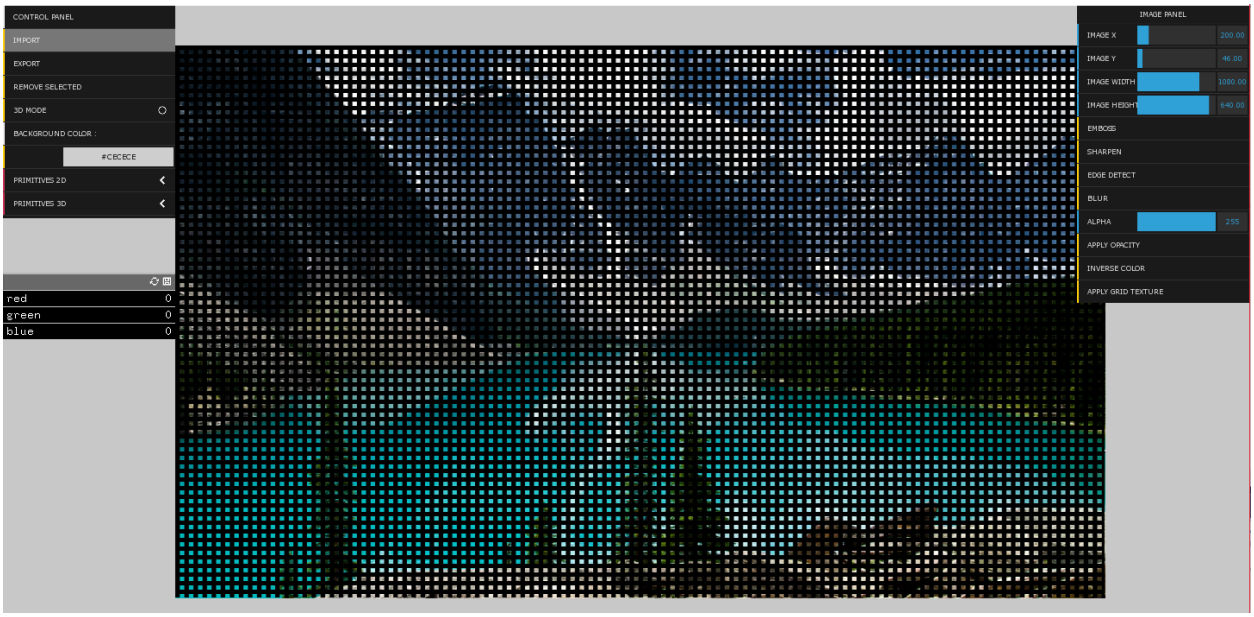
6.5.2.3 Filtre de flou sur une image de paysage.



6.5.3 Texture procédural

Il est possible d'appliquer un quadrillage noir sur une image. Cette texture est créée de façon procédurale avec un algorithme. Il est possible d'appliquer cette texture à l'aide du bouton "Apply grid texture" dans le panneau de contrôle d'une image. Dans l'image qui suit, un grillage est appliqué sur l'image de paysage.

6.5.3.1 Texture de quadrillage sur une image de paysage.



6.6 Caméra

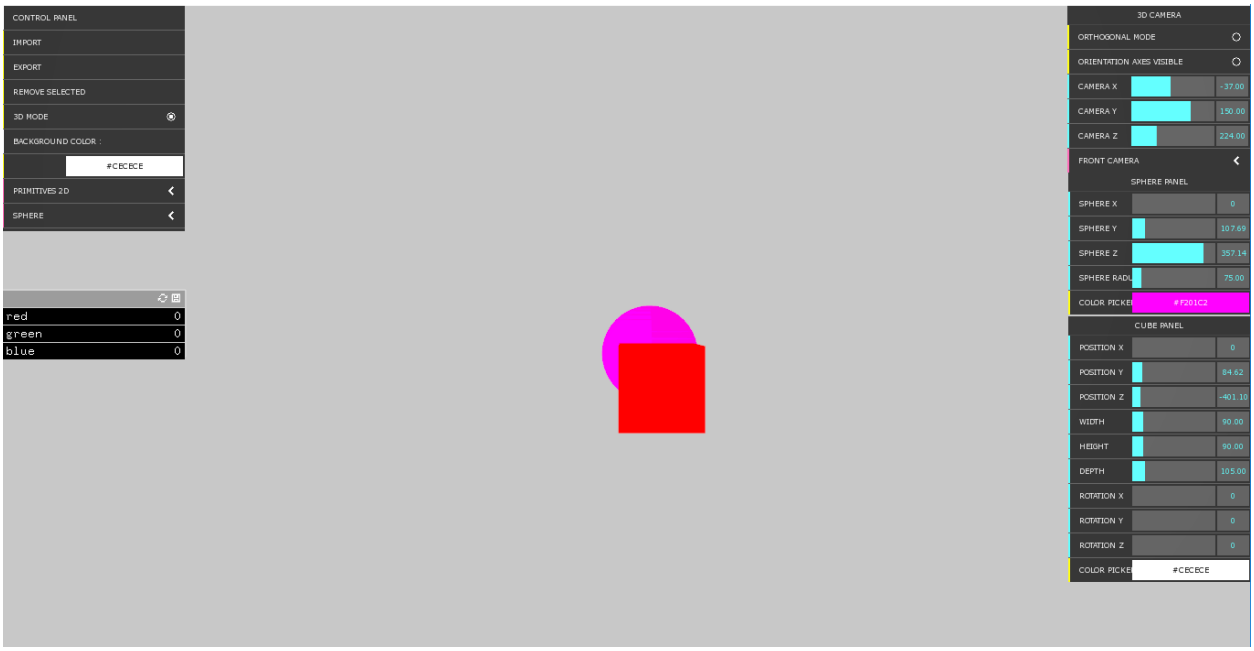
6.6.1 Point de vue

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. Pour ce faire, l'utilisateur doit sélectionner le mode 3D et ajouter des primitive 3D. Ainsi, un panneau de control de caméra est disponible au coin supérieur droit et permet de modifier la position de la caméra ainsi que son mode de projection en utilisant les composantes de l'interface utilisateur .

6.6.2 Mode de projection

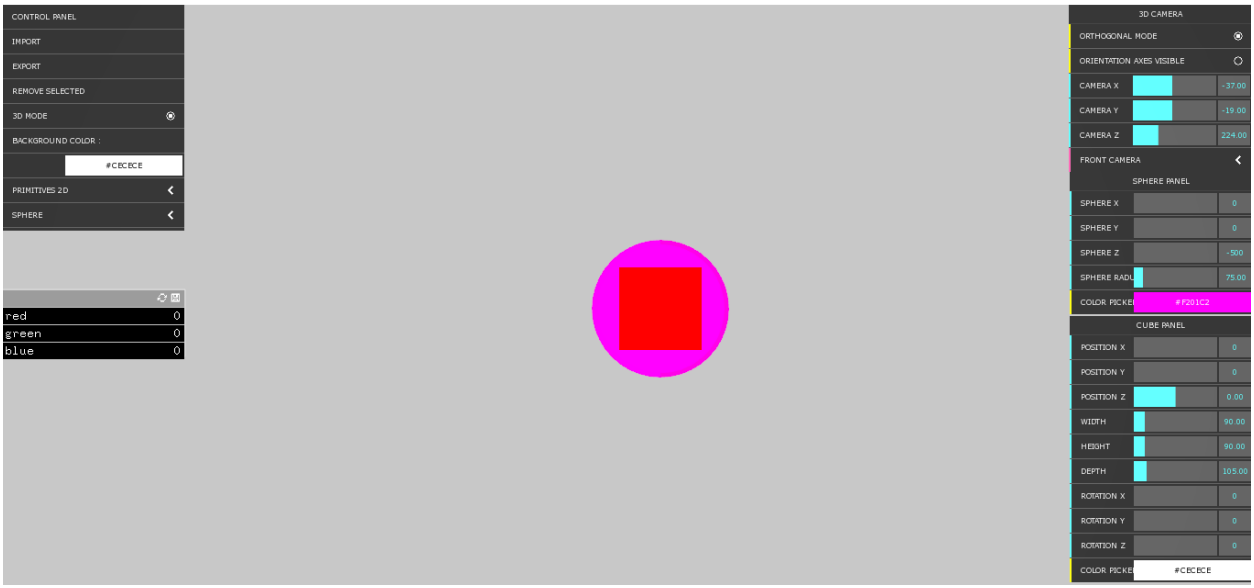
Deux type de mode de projection sont disponible pour l'utilisateur. En effet, par défaut, le mode en perspective est activé. Celui-ci utilise comme position centrale le centre du canvas, là où tous les modèles 3D sont instanciés.

6.6.2.1 Camera en mode perspective avec une sphere en arrière d'un cube.



Par la suite, si l'utilisateur désire mettre la caméra en mode orthogonal, il n'a qu'à cocher dans le panneau de contrôle de la caméra l'option "Mode Orthogonal". De cette façon, on peut voir les composantes de la scène en mode orthogonal. Afin de bien voir l'effet, il est recommandé d'instancier 2 primitives 3D ou plus.

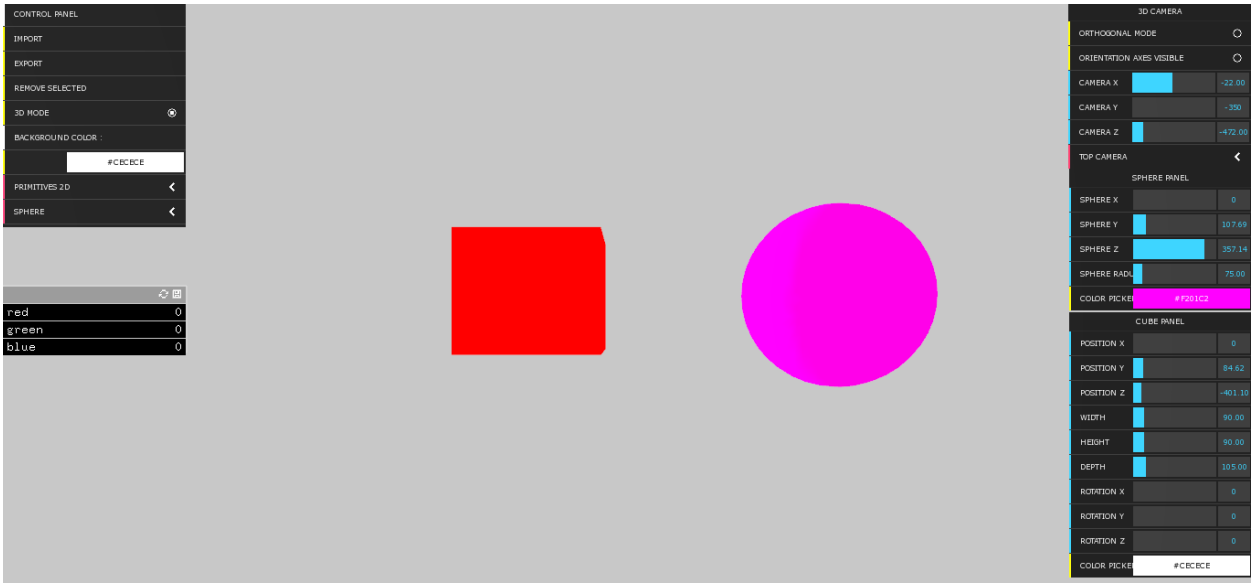
6.6.2.2 Camera en mode orthogonal avec une sphere en arrière d’un cube.



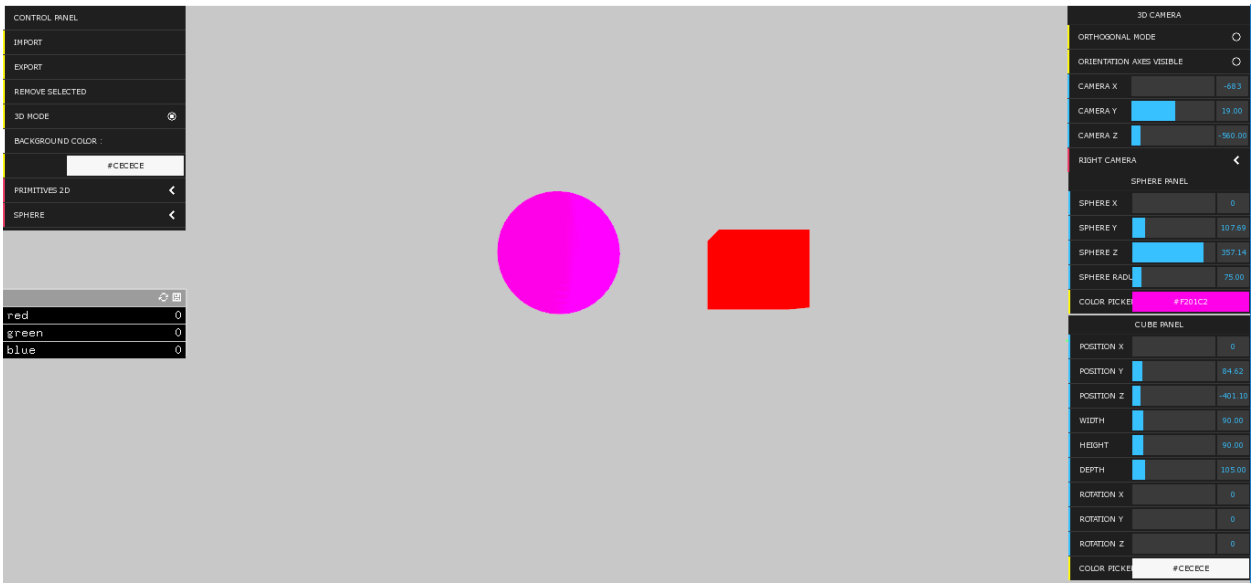
6.6.3 Agencement

Il est possible de voir une scène de plusieurs points de vue différents en pleine fenêtre d’affichage dans la fenêtre principale de l’application. La vue par défaut est la vue de face. Il est possible de changer la vue avec le menu déroulant disponible dans le panneau de contrôle de la caméra.

6.6.2.2 Camera en mode perspective avec une sphere en arrière d’un cube positionnée en haut.



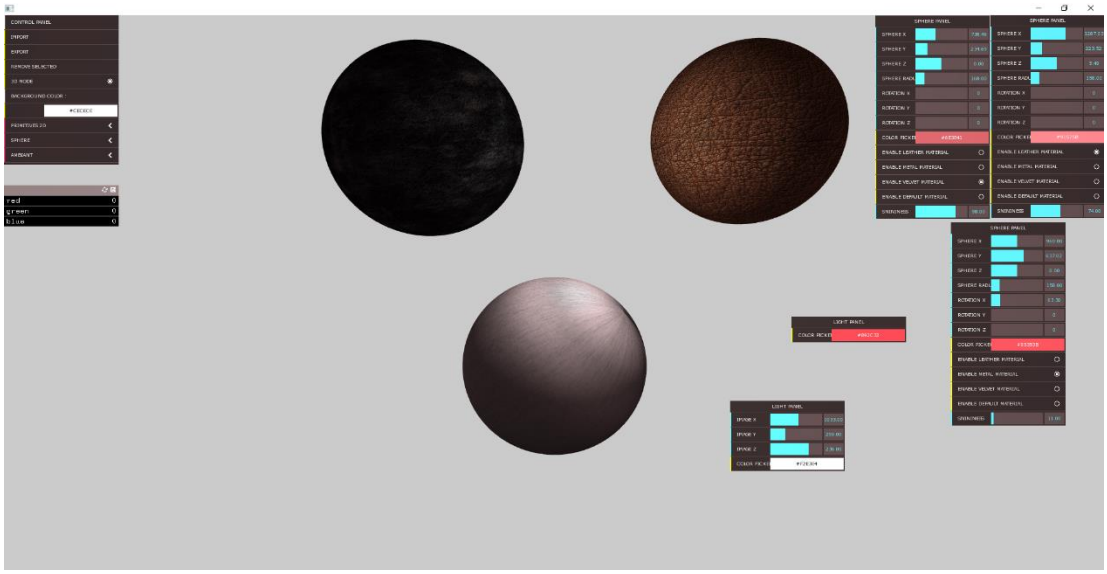
6.6.2.2 Camera en mode perspective avec une sphere en arrière d’un cube positionnée en droite.



6.7 Illumination

6.7.2 Matériaux

6.7.2.1 Trois sphères avec les différents matériaux.



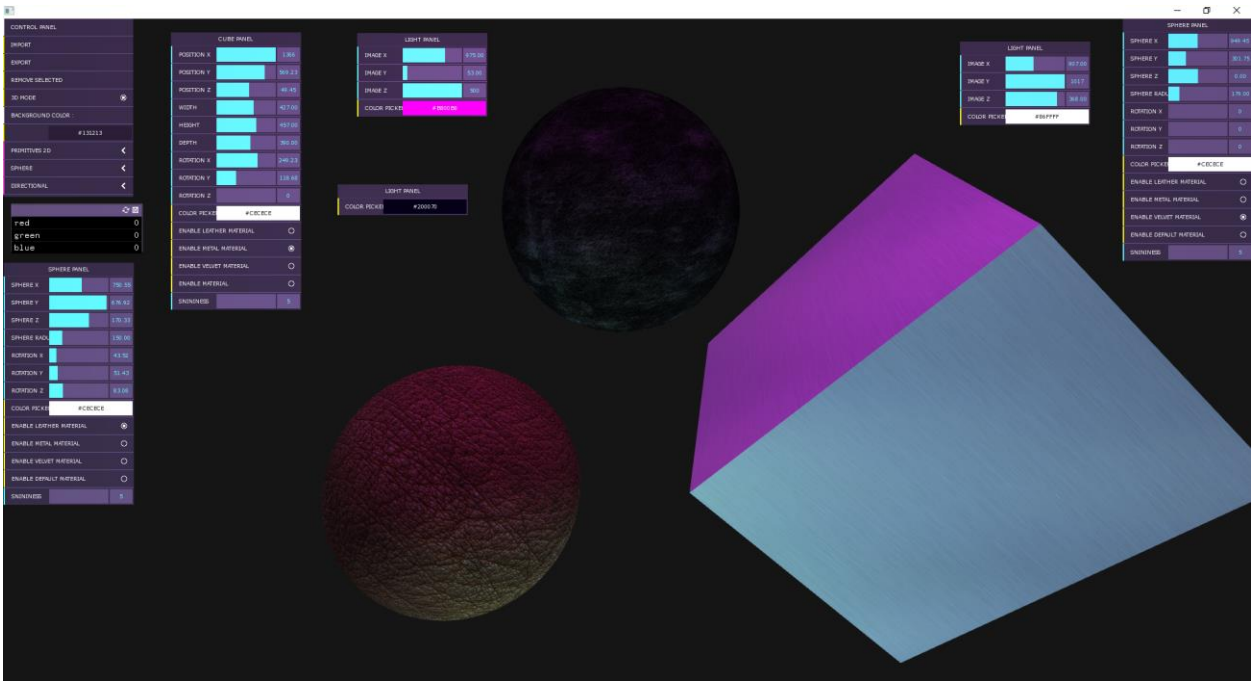
Pour certaines primitives 3D (cube et sphères), il est possible de choisir parmis trois matériaux différents : Cuir, velvet et métal. Chaque matériel possède un niveau de réflexion différent afin d'offrir plus de réalisme. Il est aussi possible de ne pas sélectionner de texture réaliste et de seulement appliquer une couleur et un niveau de réflexion à la primitive 3D afin de créer un matériel sur mesure.

6.7.3 Type de lumière

Dû à la nature des lumières dans notre programme, il est possible d'ajouter autant de lumières de chacune des quatres types demandés (ambiante, directionnelle, ponctuelle et projecteur). Chaque lumière, sauf ambiante, peut être positionnées et changée de couleur afin d'ajuster l'illumination de la scène, l'effet donné par les lumières et de les distinguer.

6.7.5 Lumière multiples

6.7.5.1 Différentes lumières appliquées à une scène 3D.

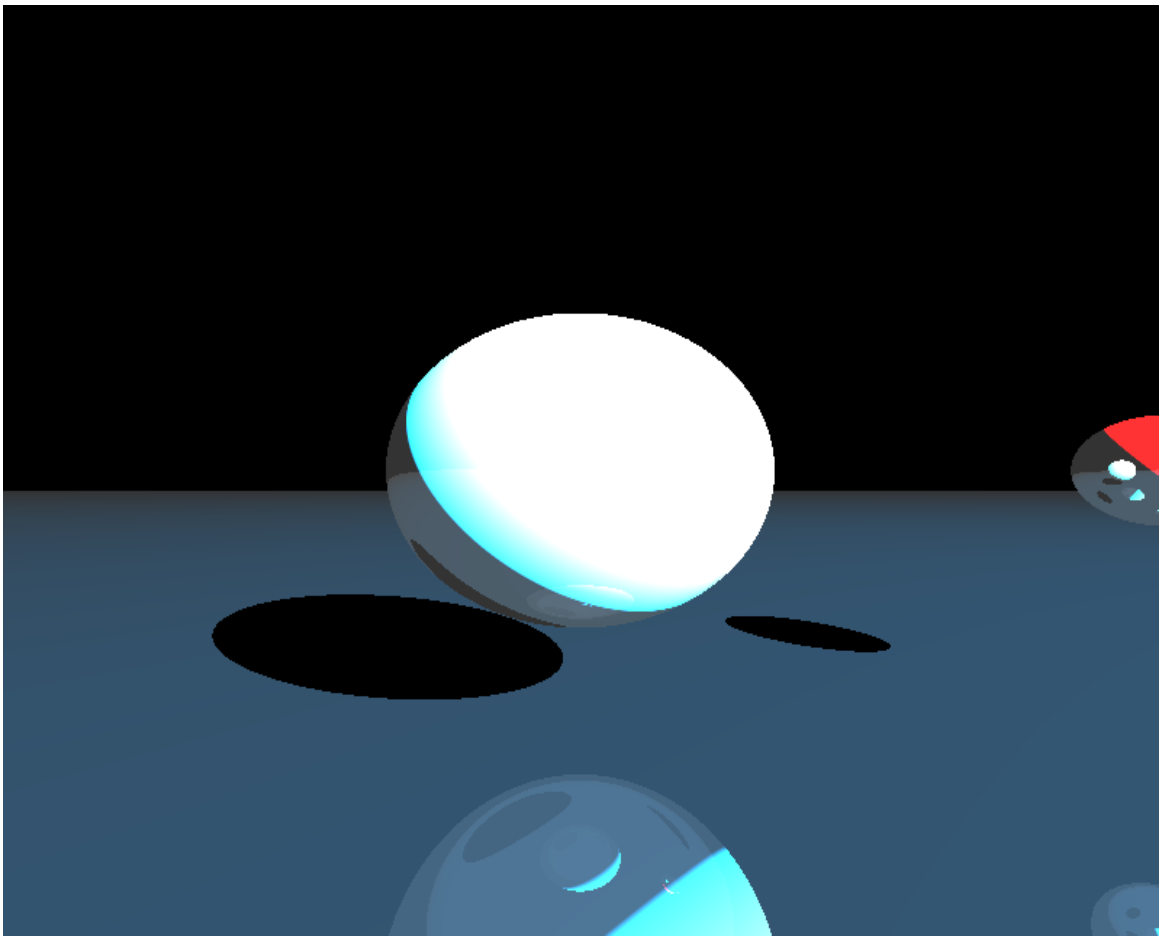


Il est possible d'ajouter 4 lumières différentes ou plus à une scène contenant des objets. Ces lumières affectent les objets selon leur niveau de réflexion mais aussi avec leur couleur, position et atténuation selon leur distance. Chaque type de lumière affecte les objets selon son type.

6.8 Lancer de rayons

Avant d'activer le mode lancer de rayons dans le panneau de contrôle, il faut d'abord rendre des éléments dans la scène. Lorsque tous les éléments 3D sont placés, il est possible de cliquer sur le bouton pour aller dans le mode "Lancer de rayons" : une nouvelle fenêtre s'ouvrira et dévoilera la scène en utilisant le lancer de rayons.

6.8.1 Deux sphères dans un mode "Lancer de rayons".



6.8.2 Réflexion

Le lancé de rayon implémenté dans notre application permet d'obtenir des réflexions comme celles créées par un miroir en reflétant une image de l'objet et pouvant la refléter à l'infini lorsque deux miroirs sont face à face.

6.8.4 Ombrage

Le lancé de rayon entre en contacte avec les objets présents dans la scène et crée des ombres en prenant compte de l'angle de la lumière et la taille de l'objet ainsi que les objets devant eux, créant des ombres réalistes.

6.8.5 Illumination globale

Le rendu de la scène avec une illumination globale présente la couleur des objets seulement du côté exposé à la lumière et laisse la partie cachée des objets ombragée, donnant un effet réaliste à la scène lors du rendu.

6.9 Topologie

6.9.1 Courbe cubique

Il est possible de rendre deux types de courbes cubiques avec 4 points de contrôle. Il y a la courbe de Bézier cubique et la courbe d'Hermite cubique. En instanciant chacune de ces courbes, un panneau de sélection apparaît pour modifier la grosseur du trait et la couleur de la courbe. Il y a aussi un onglet déroulant pour choisir le numéro d'un des points de contrôle et de par la suite modifier ses coordonnées en X, Y ou Z. Chacune de ces courbes peut être rendue dans le mode 2D ou le mode 3D du logiciel. Il est possible d'afficher ou de cacher les points de contrôle.

6.9.1.1 Courbe de Bézier cubique en bleu et courbe d'Hermite en rose avec sensiblement les même points de contrôle (2D).

6.9.1.2 Courbe de Bézier cubique en violet et courbe d’Hermite en orange (3D).

6.9.2 Courbe paramétrique

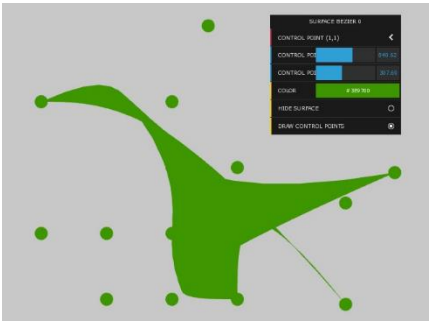
Il est possible de rendre une courbe paramétrique. Il s’agit de la courbe de Bézier à 6 points de contrôle. À l’instanciation, un panneau de contrôle apparaît pour changer l’apparence de la courbe, soit sa couleur et la grosseur du trait. Il y a aussi un onglet déroulant pour choisir un point de contrôle en particulier pour pouvoir modifier ses coordonnées. La courbe de Bézier à 6 points de contrôle peut être rendu en 2D et en 3D. Il est possible d’afficher ou de cacher les points de contrôle.

6.9.2.1 Courbe de Bézier paramétrique à 6 points de contrôle (2D).

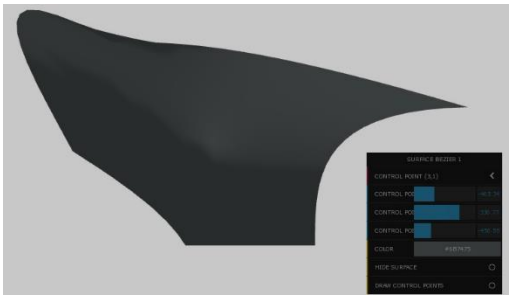
6.9.3 Surface paramétrique

Il est possible de rendre une surface paramétrique, soit une surface de Bézier bicubique. Cette surface possède 16 points de contrôle positionnés sous forme de 4 par 4. Un panneau de contrôle permet de modifier les coordonnées des points de contrôle, ainsi que la couleur de la surface. Cette surface peut être rendue autant dans le mode 2D que le mode 3D du logiciel. Il est possible d’afficher ou de cacher les points de contrôle, ainsi que la surface.

6.9.3.1 Surface de Bézier bicubique avec ses points de contrôle (2D).



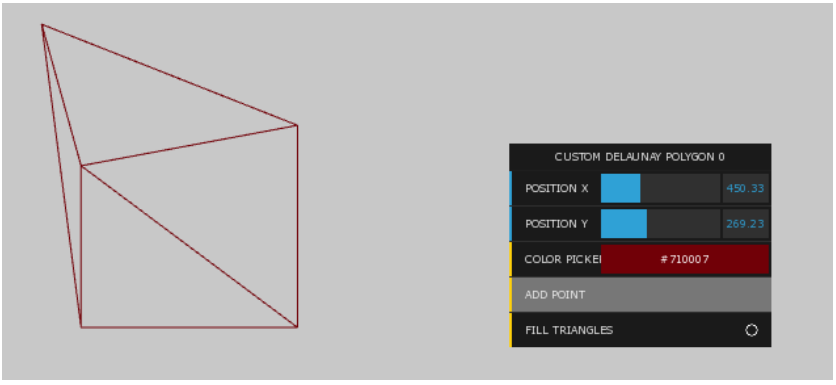
6.9.3.1 Surface de Bézier bicubique sans ses points de contrôle (3D).



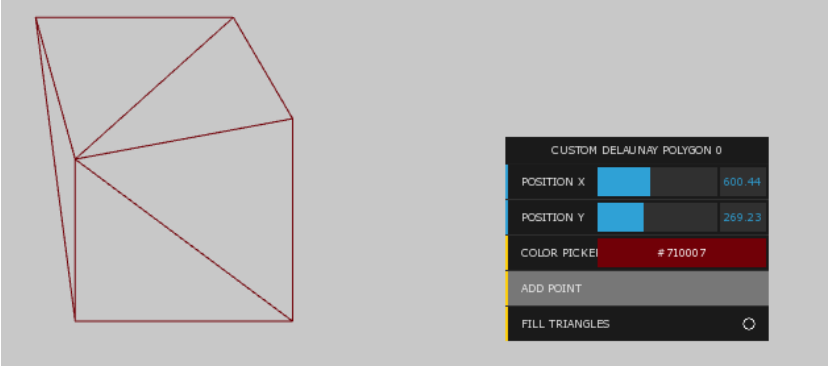
6.9.5 Triangulation

Il est possible de générer un maillage triangulaire à partir d'un ensemble de sommets à l'aide de la triangulation de Delaunay. À l'instanciation, il n'y a rien qui apparaît. Il faut modifier les valeurs en X et en Y (et en Z si en mode 3D), puis cliquer sur le bouton "Add point". Lorsqu'il y aura trois points, le triangle apparaîtra. En ajoutant d'autres points, l'algorithmes de Delaunay agira pour refabriquer le maillage triangulaire.

6.9.5.1 Maillage triangulaire fabriqué avec la triangulation de Delaunay avec 5 points.



6.9.5.2 Même maillage triangulaire que l'image 6.9.5.1, mais avec un point de plus et des arêtes modifiées.

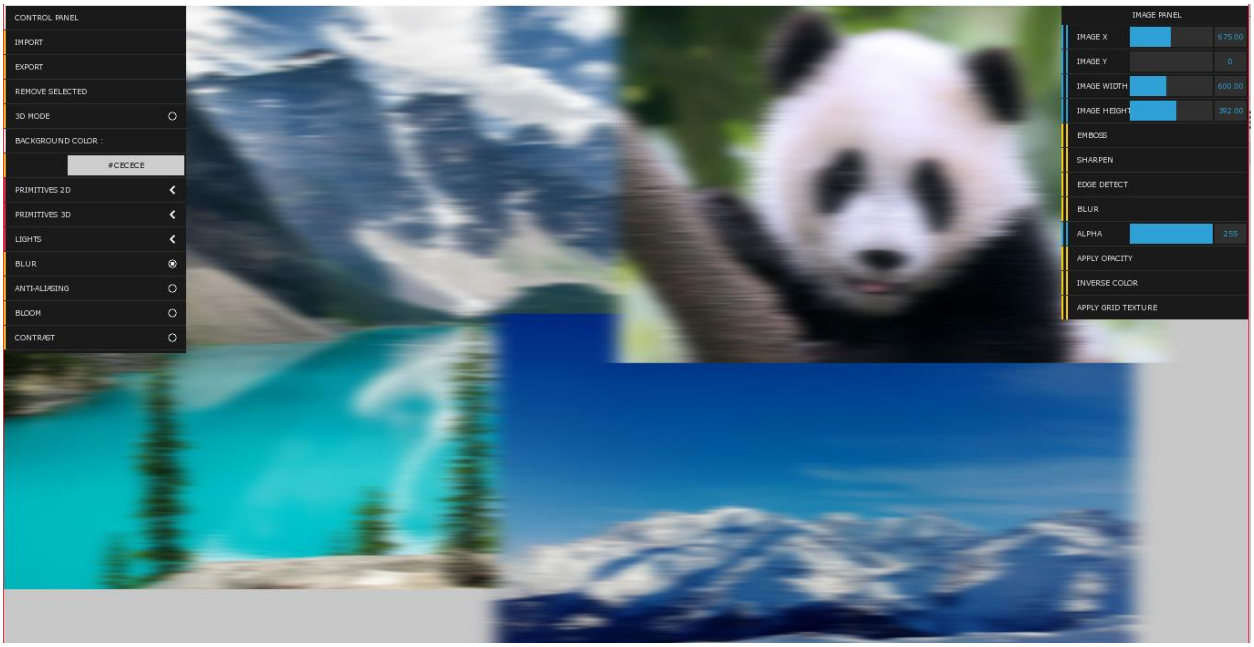


6.10 Techniques de rendu

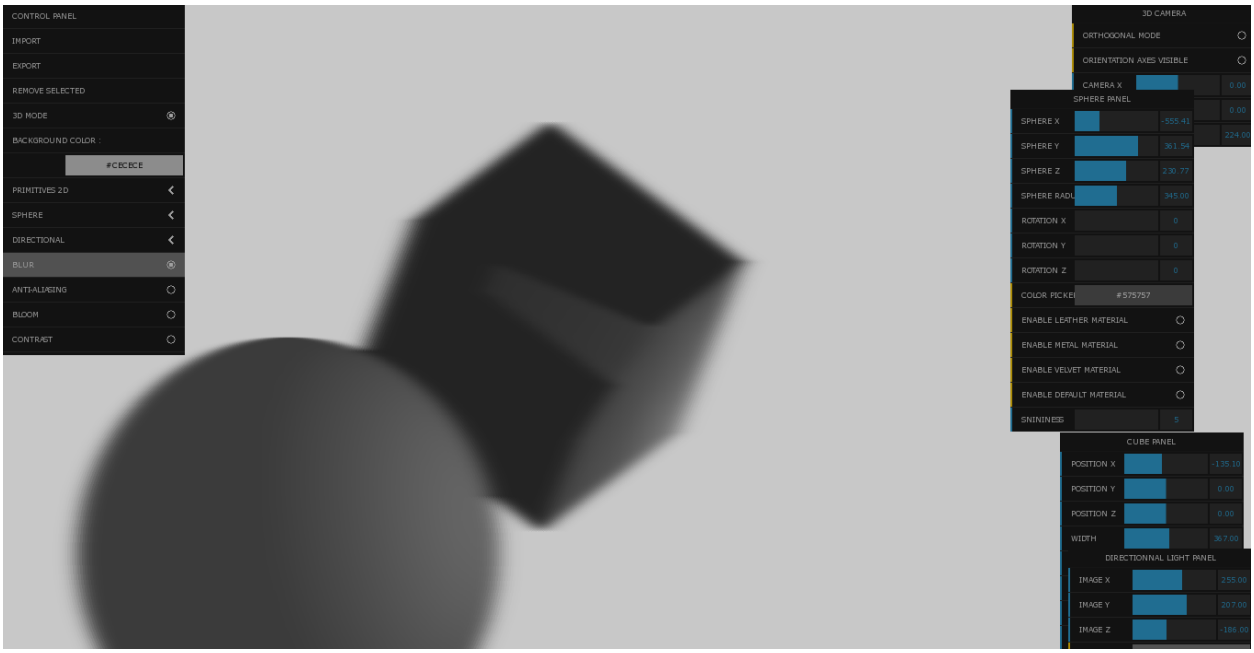
6.10.1 Effet en pleine fenêtre

Il est possible d'appliquer trois types d'effet visuel sur toute la surface de la fenêtre d'affichage, autant pour le 2D que pour le 3D. Le premier effet est un flou :

6.10.1.1 Flou appliqué sur des images (2D)



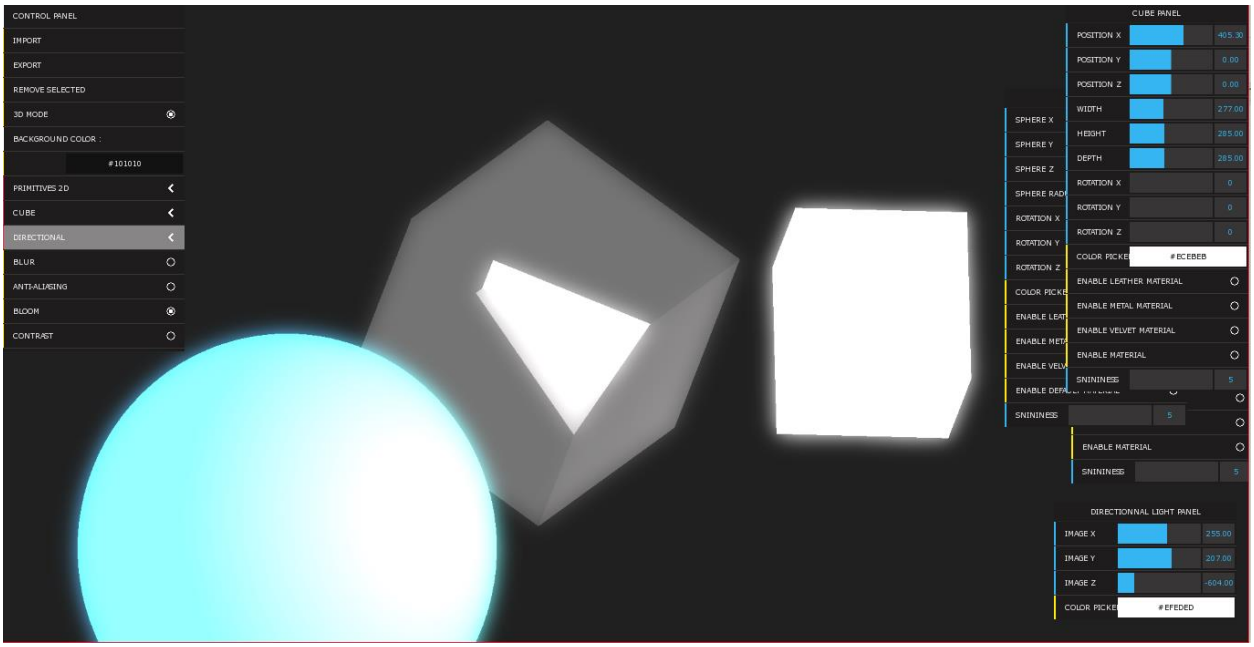
6.10.1.2 Flou appliqué avec deux modèles(3D)



Le deuxième effet est le “Bloom”, aussi appelé le flou lumineux. Cet effet fait ressortir les zones claires et lumineuses.

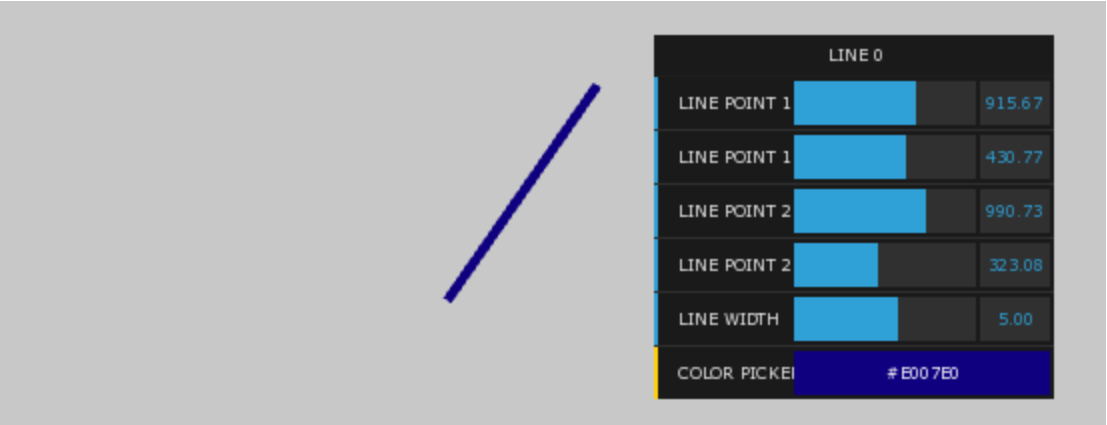
6.10.1.3 Bloom appliqué sur des images (2D)

6.10.1.4 Bloom appliqué avec trois modèles(3D)

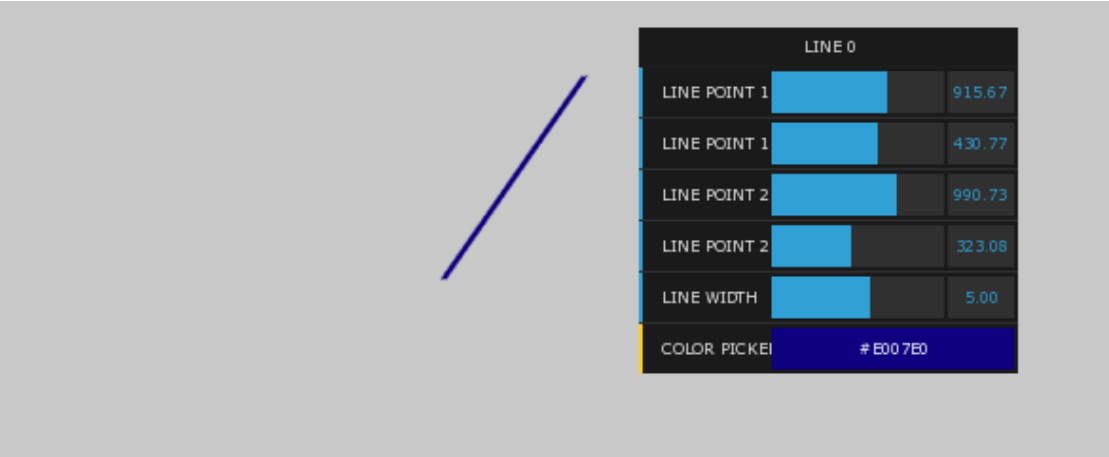


Le troisième est l'anticrénelage. Cet effet visuel permet de diminuer l'effet d'escalier sur les contours des formes sur les images et les modèles. Les bords sont adoucis pour enlever le crénelage.

6.10.1.5 Avant l'appliquation de l'anticrénelage sur une primitive (2D)



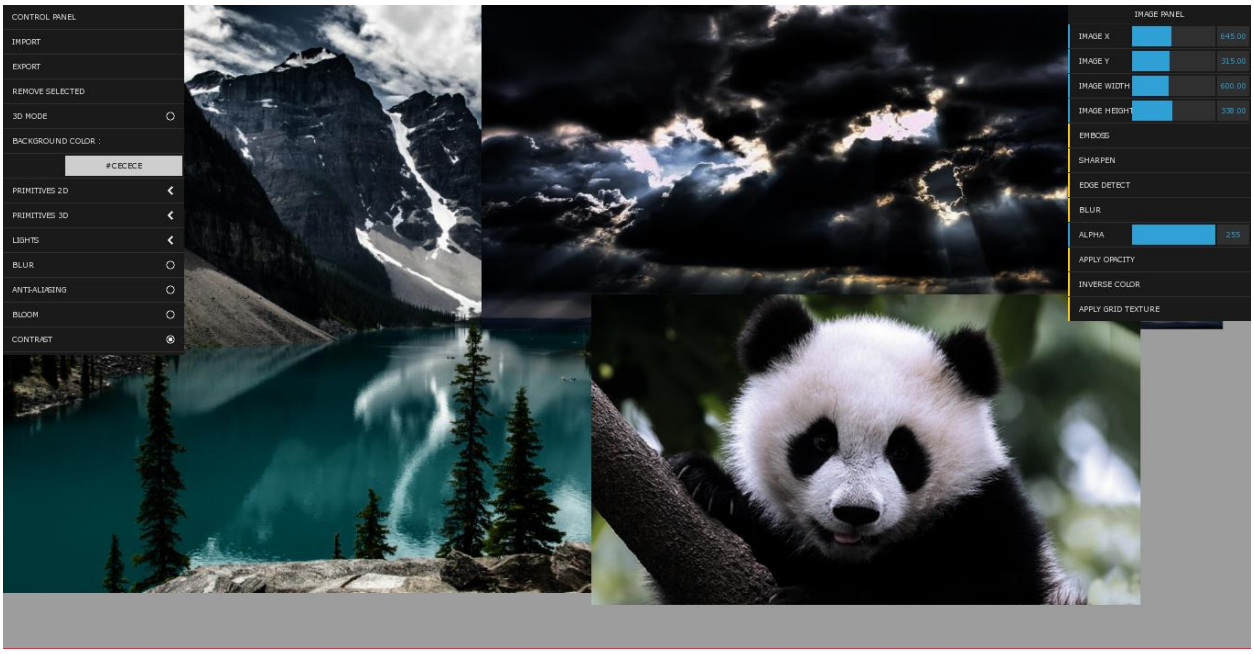
6.10.1.6 Après l’appliquation de l’anticrénelage sur une primitive (2D)



6.10.2 Style libre

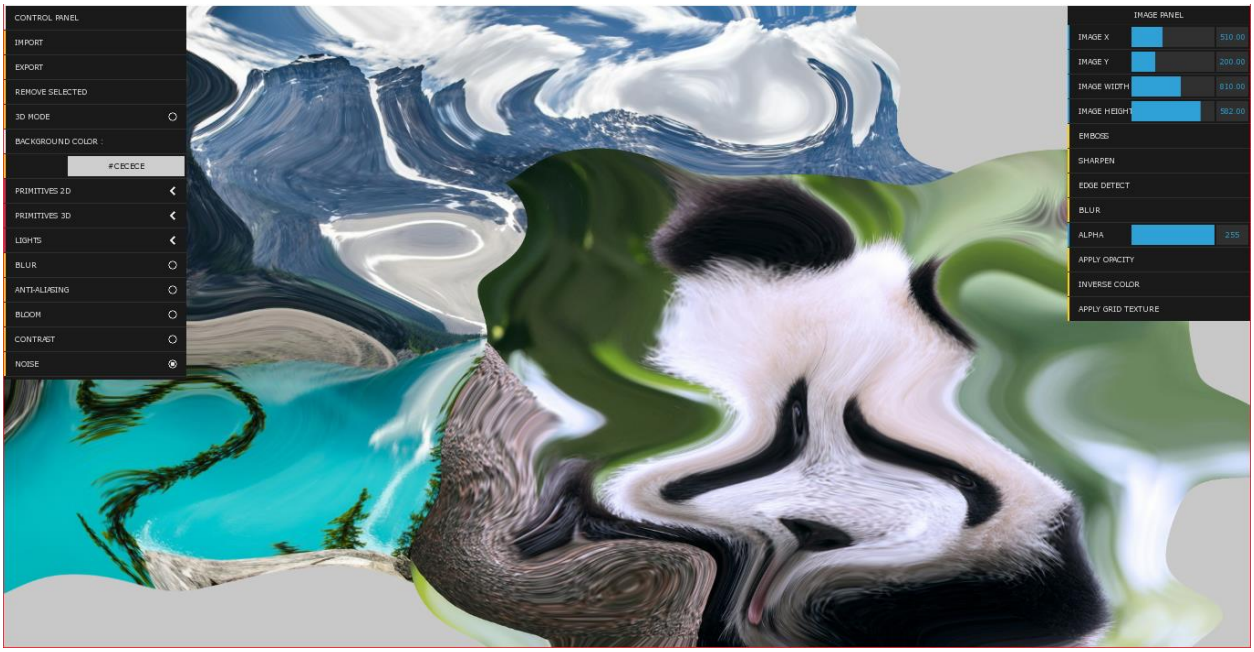
Pour le style libre, il est possible de réaliser deux effets visuels sur les images et les modèles à l’écran. Le premier change le contraste des couleurs de tous ce qui est présent à l’écran.

6.10.2.1 Application d’un effet de contraste sur des images (2D)

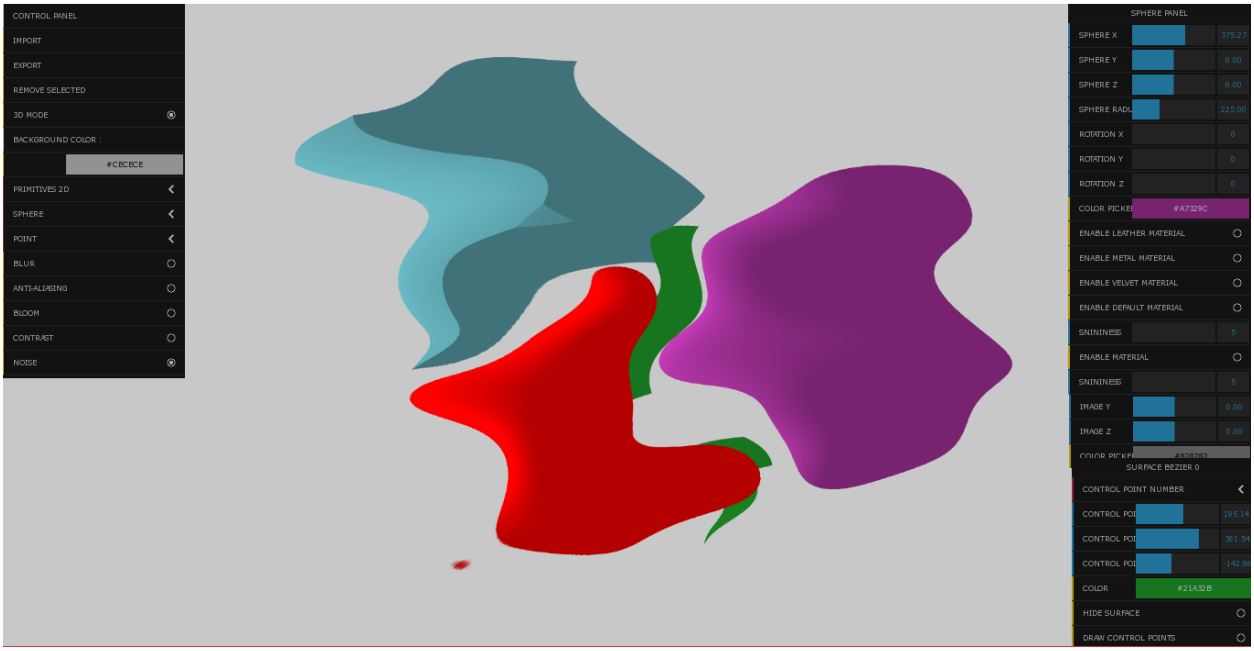


Le deuxième effet visuel est un bruit graphique qui déforme ce qui est à l'écran en faisant tout bouger aléatoirement. Les modèles et les images sont en mouvement tant que l'effet de bruit est activé.

6.10.2.2 Application d'un effet de bruit sur des images (2D)



6.10.2.3 Application d'un effet de bruit sur des modèles (3D)



7. Ressources

Non-originale :

1. Répertoire de la librairie ofxDatGui : https://github.com/braitsch/ofxDatGui
2. Répertoire de la librairie ofxBezierSurface : https://github.com/charlesveasey/ofxBezierSurface
3. Répertoire de la librairie ofxPostProcessing: https://github.com/neilmendoza/ofxPostProcessing/tree/stable
4. Répertoire de la librairie ofxDelaunay: https://github.com/obviousjim/ofxDelaunay
5. Image de panda : http://kids.sandiegozoo.org/sites/default/files/2017-04/animal-hero-panda3.jpg
6. Image de paysage : https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS04w4PZoXQGQdKXKfJjy5XIinMGIYg81K_r-2eIYZxgBburI9NYw
7. Image d'un paysage de ciel : http://aws-cf.imdoc.fr/prod/photos/0/8/7/7362087/5980126/big-5980126bf0.jpg?v=5
8. Image d'une montagne hivernale : http://www.grainedesportive.fr/blog/wp-content/uploads/2013/01/montagne-paysage.jpg
9. Modèle de triceratops : https://free3d.com/3d-model/puo-3916-78869.html
10. Diagramme MVC : https://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur

Originale :

Les autres images sont des captures d'écran de l'interface de l'application programmée dans le cadre de ce cours

8. Présentation

Damien Arroyo

Je suis un étudiant de 3ème année en Génie Logiciel. J'ai quelques bases en infographie et en programmation de jeux de jeux vidéo lors de mes cours de programmation au cégep ainsi qu'avec des projets personnels. Durant ma dernière session de cégep, nous avons monté notre propre "engine" 3D pour le projet principal étant in jeux "rogue-like" 3D. Le projet personnel le plus relié à l'infographie était une base de jeux 2D inspiré de Nuclear Throne.

Je prend ce cours afin de pousser mes connaissances en infographie ainsi que pour suivre le cours de conception de jeux vidéo. De plus, je suis intéressé à jouer avec la réalité virtuel soit dans des cours ou dans des projets personnels.

Marc-André Tremblay

Je m'appel Marc-André Tremblay et je suis à ma troisième années du bac en informatique. Je n'ai aucune base en infographie, alors je suis très motivé à en apprendre sur le sujet.

Je suis ce cours parce que j'envisage de peut-être me diriger vers le domaine des jeux vidéo et ce cours est un préalable pour le cours de jeux vidéo. Je veux en apprendre davantage sur la programmation graphique et être capable de faire des modèle 3D.

Marcel Samson-Morasse

Je m'appelle Marcel Samson Morasse et je suis étudiant en 3ème année de Génie logiciel. J'ai quelques bases en infographie puisque j'ai fait le cours de dessin pour ingénieur et que j'ai travaillé avec Unity lors d'un de mes stages.

Je prends ce cours puisqu'il est préalable au cours de conception de jeux vidéos et que j'aimerais bien avoir une carrière dans ce domaine un jour.

Rémy Tremblay

Je suis étudiant en 3ème année en Génie logiciel. J'aime le domaine des jeux vidéos et je souhaite ainsi faire le cours de jeux vidéos à l'automne à la suite du cours d'infographie. Ayant suivi un cours de photographie scientifique à l'Université du

Québec à Trois-Rivières, j’avais déjà exploré en surface quelques techniques d’infographie, soit les différentes techniques d’encryption d’images et le fonctionnement de différents filtres utilisant des kernels.