Travail pratique 1

Travail présenté à M. Philippe Voyer dans le cadre du cours :

IFT-3100 – Infographie

Travail réalisé par :

Gabriel Bergeron 536 888 644

Mathieu Faucher 536 890 054

David Duchesne 536 895 884

8 mars 2023

1 Sommaire

Le but de notre projet est de reproduire une scène d'un jeu « platform fighter » et de pouvoir construire des niveaux. Notre projet séparera donc l'écran de travail en deux avec, pour la vue du haut, une vue globale et fixe du niveau en question et, pour la vue du bas, une caméra qui peut se déplacer librement dans l'espace 3D. Il faudra donc pouvoir créer des plateformes et des obstacles ayant n'importe quelle géométrie. La vue globale et fixe pourra ajouter des objets 2D comme des images et des primitives vectorielles qui auront une certaine profondeur. Celles-ci pourraient par exemple faire office d'image d'arrière plan. Pour être en mesure d'offrir une expérience utilisateur agréable, l'interface est composé d'éléments faits sur mesure pour donner un aspect plus crédible à l'application.

Il est possible de voir deux vues qui seraient possible d'observer dans la scène du haut et dans la scène du bas.

Figure 1: Exemple scène du haut

Figure 2: Exemple scène du haut

2 Interactivité

L'application a 5 curseurs, un curseur normal, un curseur pour dessiner, un curseur de rotation, un curseur de translation et un curseur de changement de proportion. En appuyant sur le bouton « File » dans la bar de titre, il est possible d'importer une image pour ensuite la dessiner en mode « Drawing ». Il est également possible d'exporter les deux scènes en format png. De plus, l'application possède un graphe de scène qui permet de grouper des éléments avec une arborescence de parents-enfants. Il n'y a qu'à prendre les éléments dans le graphe avec la souris et les placer sur un autre pour les rendre enfants du second. Il y a aussi un outil de dessin qui permet de choisir une forme ainsi que des paramètres pour l'épaisseur de ligne, la couleur de ligne et la couleur du remplissage si applicable. Il suffit d'entrer en mode « Drawing » à partir de la bar de titre et de sélectionner les paramètre désirés. Il ne reste qu'à dessiner dans la zone de dessin et les formes devraient apparaître. Si des éléments de la scène sont sélectionnés à partir du graphe de scène, il est possible d'effectuer des transformations sur ceux-ci en entrant en mode « Transform » et en choisissant le type de transformation ainsi que les axes affectés dans les deux panneaux en bas à gauche. Si un parent est sélectionné, tous ses enfants subissent la même transformation. Il est aussi possible de faire afficher un histogramme en appuyant sur le bouton de la bar de titre « Show ». Ils s'affichent alors par-dessus chacune des scènes.

3 Technologie

Pour ce projet, nous avons décidé de rester simples. Nous avons donc utilisé visual studio sous Windows avec Openframeworks sans librairie externe. De plus, nous utilisons les addons ofxAssimpLoader et ofxGui. Pour le contrôle de version, GitHub est utilisé et il est paramétré pour nécessiter une approbation d'un autre membre de l'équipe pour merge sur la branche main. Les taches sont attribuées et suivies avec Trello.

4 Compilation

Pour compiler notre projet, il faut avoir Visual Studio et OpenFrameworks d'installés. La version de C++ utilisée est C++ 14. Ensuite, il faut cloner notre répertoire de code disponible ici : https://github.com/zeckrust/RealEngine dans le dossier apps d'OpenFrameworks. Finalement, il faut build la solution et un exécutable devrait être généré.

5 Architecture Logicielle

L'architecture de notre application est principalement basée autour de deux singletons : le Renderer et le Gui. Le Renderer possède un pointeur du Gui, le curseur et les deux instances de dessin permettant d'afficher le contenu des deux scènes. Il coordonne alors l'affichage de ces modules.

Puisque les informations permettant de dessiner et de transformer les scènes se retrouvent dans le Gui, il a été jugé une bonne décision de rendre le Gui un singleton. Il est alors possible d'accéder aux informations depuis n'importe quelle classe qui possède le pointeur de l'instance du Gui.

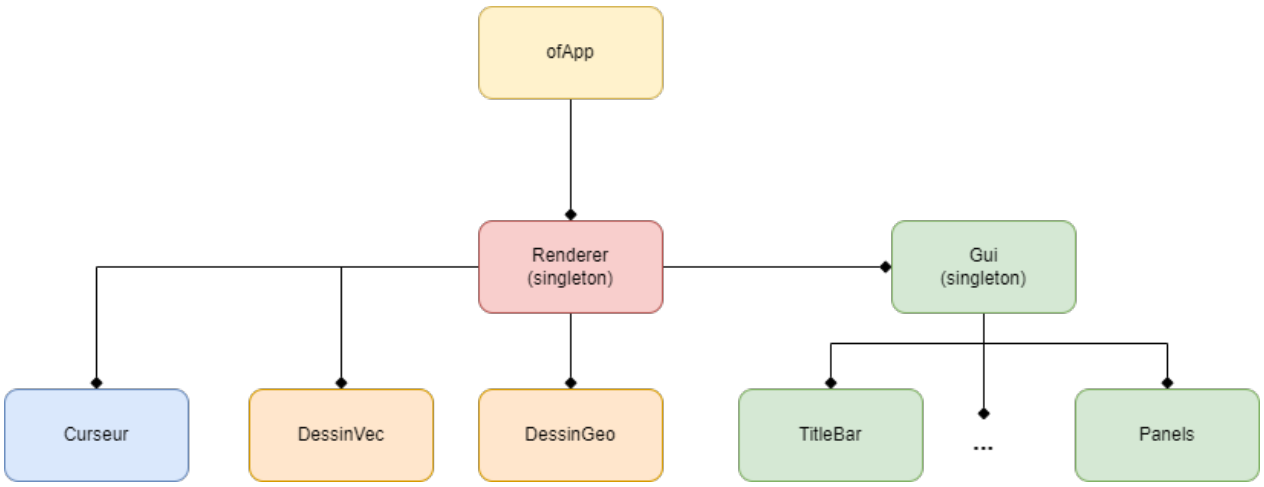

Figure 3: Architecture logicielle globale

6 Fonctionnalités

6.1 Image

6.1.1 Importation d'image

Pour que l'importation d'images fonctionne, il a fallu ajouter un bouton d'importation dans notre GUI. Ensuite il a fallu que quand le bouton est actionné, un explorateur de fichier ouvre et permette de choisir une image. Cette image est ajoutée à une classe d'objet vectoriel qui possède une matrice de transformation, initialisée à la matrice identité, une position et une dimension. L'image est ensuite rendue selon ces attributs et la caméra.

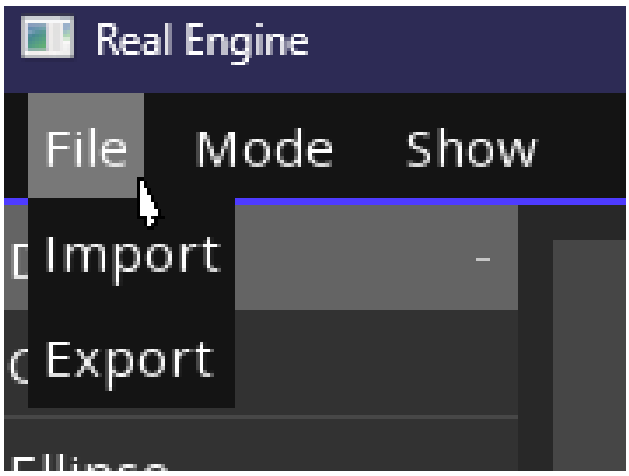

Figure 4: Bouton d'importation

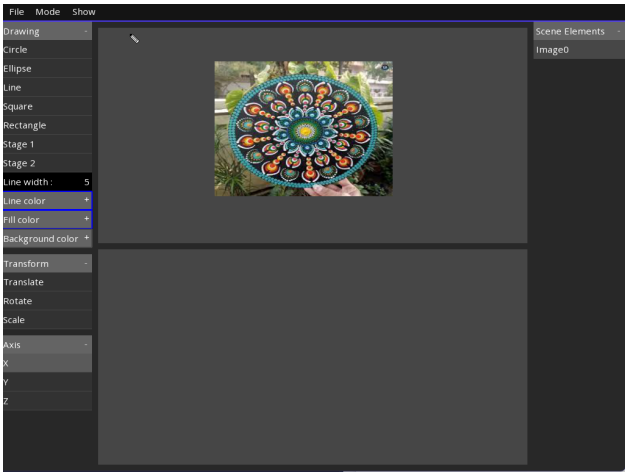

Figure 5: Dessin d'image importé

6.1.2 Exportation d'image

L'exportation fonctionne en prenant une copie des deux framebuffers et en les exportant en PNG dans le dossier sélectionné lors de l'appui du bouton d'exportation.

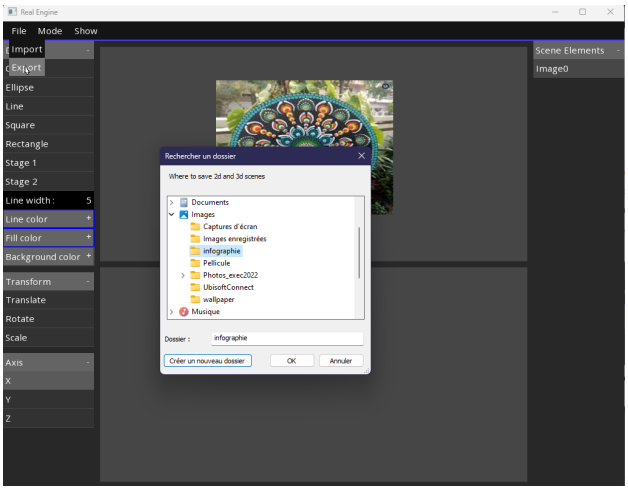

Figure 6: Exemple de choix de dossier

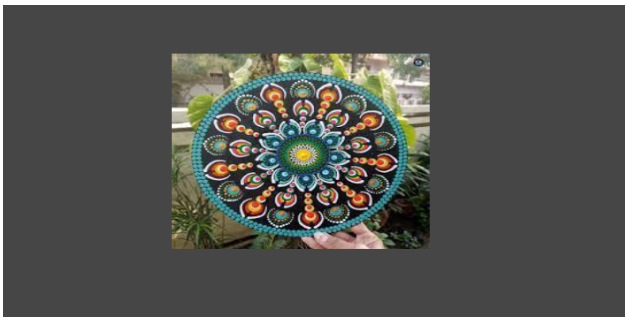

Figure 7: Image exporté

6.1.3 Échantillonage d'image

Pourrait être ajouté dans une prochaine version.

6.1.4 Espace couleur

L'interface supporte l'espace de couleur RGB ainsi que HSB, une image vaut mille mots!

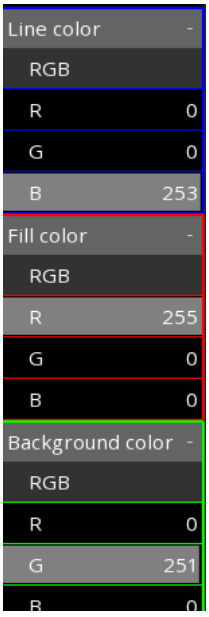

Figure 8: Exemple de choix de couleurs en RGB

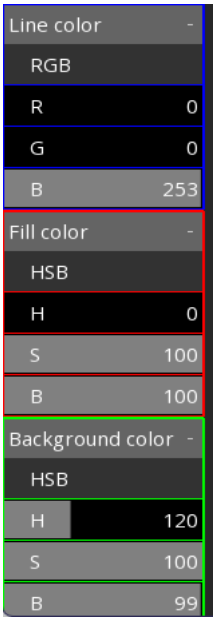

Figure 9: Les deux choix de couleurs du bas ont été changé pour HSB

6.1.5 Histogramme

L'histogramme est activé avec Show->Histogram. Il calcul la valeur des pixels de chacun des framebuffers et s'affiche par dessus la zone de dessin.

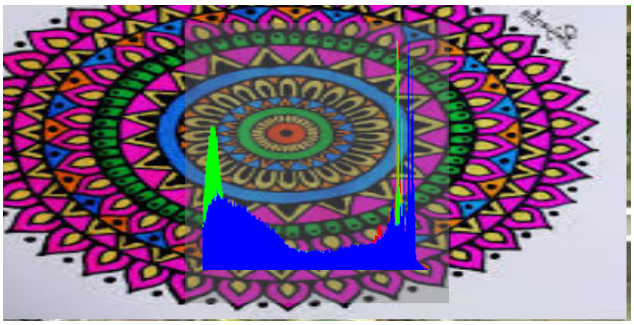

Figure 10: Exemple d'histogramme avec mandala

6.2 Dessin vectoriel

6.2.1 Curseur dynamique

5 curseurs ont été dessinés, ils sont donc propre à notre projet. Ceux-ci changent selon le contexte. Si la sourie se trouve au dessus d'une zone de dessin, le curseur prend la forme d'un crayon. Celui-ci prend la forme d'une main pour intéragir avec la proportion. Les curseurs de rotation et de proportionnalité sont utilisés pour les transformations vectorielles. Enfin, le curseur de base est utilisé pour le reste de l'application.

6.2.2 Outils de dessin

Sur un paneau à gauche, les options d'outils de dessins sont ajouté. Les options sont largeur de ligne, couleur de ligne, couleur de remplissage et couleur du background.

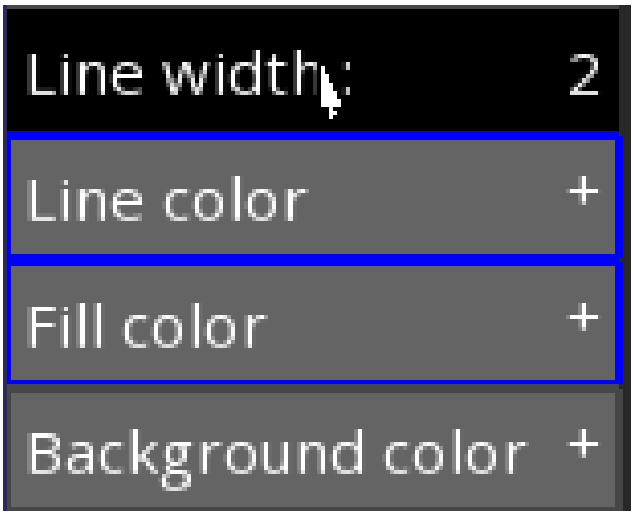

Figure 11: Capture d'écran du paneau d'outil de dessin

6.2.3 Primitives vectorielles

5 primitives verctorielles sont implémentées. On peut dessiner des lignes, des cercles, des ellipses, des carrés et des rectangles. On peut contrôler la couleur du remplissage et du contour des primitives avec les panneaux à gauche. Pour les dessiner, il suffit de sélectionner la primitive avec les boutons sur la gauche, de cliquer sur la scène et de déplacer sa sourie pour la voir se dessiner. On peut les effacer de la zone de dessin en faisant un clic droit sur le widget à droite conrrespondant et en appuyant sur "delete".

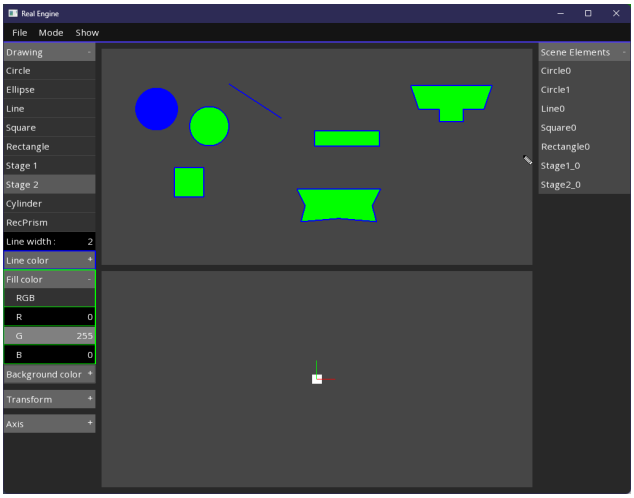

Figure 12: Capture d'écran des primitives vectorielles et des formes vectorielles

6.2.4 Formes Vectorielles

Il y a deux formes vectorielles. Elle peuvent être dessinées de la même manière que les primitives vectorielles. Puisque l'application veut permettre la visualisation de scène de jeux de combat de plateforme, les deux formes vectorielles sont inspirée de stage.

6.2.5 Interface

L'interface est complètement faite sur mesure, elle est constituée simplement de ofRectangle et de ofTrueTypeFont. Elle est complètement interactive avec des panneaux qui peuvent se minimiser, une barre-titre qui s'ouvre sur un clic et reste ouverte jusqu'à ce que la souris soit hors de la barre-titre pour plus de 2 secondes.

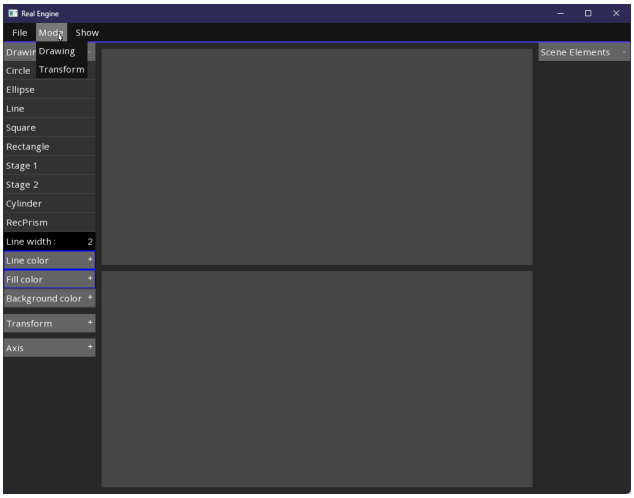

Figure 13: Capture d'écran de l'interface

6.3 Transformation

6.3.1 Graphe de Scène

Un graphe de scène placé à la droite de l'interface permet d'organiser les éléments de la scène en permettant de grouper des éléments et en permettant de sélectionner des groupes d'éléments pour la transformation. Il permet aussi de supprimer des groupes d'éléments.

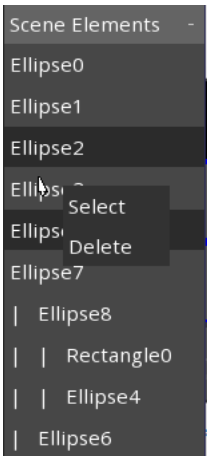

Figure 14: Capture d'écran du graphe de scène

6.3.2 Sélection multiple

Comme mentionné au dernier point, le graphe de scène permet les sélections multiples. Il est donc possible de sélectionner différents éléments de scène et de modifier certains attributs en commun tel que la position, la rotation et la proportion sur chaque axe.

6.3.3 Transformations interactives

Les transformations se font sur les éléments sélectionnés et leurs enfants du graphe de scène. Une fois la sélection faite, il fait choisir le mode Transformation en appuyant sur Mode->Transform. Il faut ensuite choisir un mode de transformation dans le panneau Transform de gauche et un ou plusieurs axes (sur lesquels appliquer les transformations) dans le panneau Axes. Les transformations en x et en z se font en trainant la souris de gauche à droite et les transformations en y se font en trainant la souris de haut en bas.

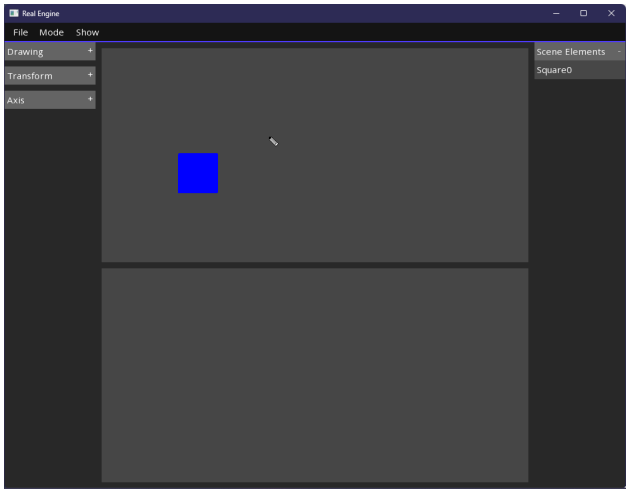

Figure 15: Capture d'écran avant la transformation

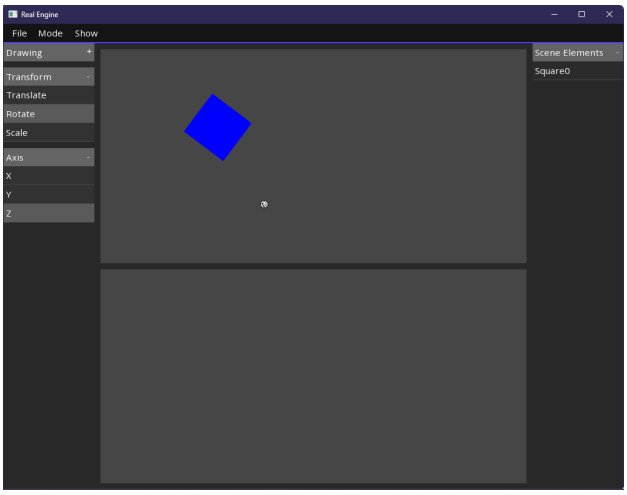

Figure 16: Capture d'écran après transformation de translation x et y, de proportion x et y et rotation en z

6.3.4 Historique de transformation

Pourrait être ajouté dans une prochaine version.

6.3.5 Système de coordonnées

N/A

6.4 Géometrie

6.4.1 Boite de delimitation

Pourrait être ajouté dans une prochaine version.

6.4.2 Primitives géometriques

Deux primitives géométriques sont implémentées, un prisme à base rectangulaire et un prisme à base octogonal. Le prisme à base rectangulaire sert de plateforme normale et le prisme à base octogonal sert plutôt de plateforme pour réapparaitre après une mort.

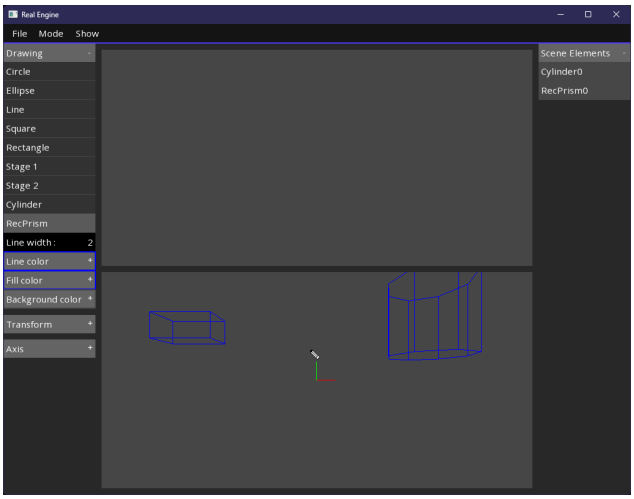

Figure 17: Capture d'écran avec nos deux primitives géometriques

6.4.3 Modèle 3D

Les modèles 3D sont presque fonctionnel, une mauvaise gestion du temps a fait que cette fonctionnalité n'est pas encore supporté.

6.4.4 Animation

N/A

6.4.5 Instanciation

N/A

6.5 Caméra

6.5.1 Caméra interactive

La caméra du bas est une caméra perspective interactive. Les touches q,w,e,y,u,i,a,s,d,h,j et k servent à faire des déplacement de la caméra. voici les fonctions des touches:

q	Avancer la caméra
w	Monter la caméra
e	Reculer la caméra
a	Déplacer la caméra à gauche
s	Baisser la caméra
d	Déplacer la caméra à droite
y	Avancer la cible de la caméra
u	Monter la cible de la caméra
i	Reculer la cible de la caméra
h	Déplacer la cible de la caméra à gauche
j	Baisser la cible de la caméra
k	Déplacer la cible de la caméra à droite

Table 1: fonctions des touches

6.5.2 Modes de projection

De par la nature de notre projet, il y a deux modes de projections dans notre projet. La vue du haut est la vue d'une caméra orthogonale et la vue du bas est la vue d'une caméra perspective.

6.5.3 Point de vue multiple

Pourrait être ajouté dans une prochaine version.

6.5.4 Occlusion

Pourrait être ajouté dans une prochaine version.

6.5.5 Portail

Pourrait être ajouté dans une prochaine version.

7 Ressource

Les seules ressources utilisées pour ce projet sont des curseurs dessinés en pixel art. Le reste de l'interface est faite avec des primitives données par OpenFrameworks.

Figure 18: curseur de base

Figure 19: curseur en forme de main

Figure 20: curseur de proportionnalité

Figure 21: curseur de dessin

Figure 22: curseur de rotation

Figure 23: curseur de sample (pas utilisé encore)

8 Présentation

8.1 Gabriel Bergeron

Je suis en troisième année de génie informatique et je suis présentement en recherche d'une branche de la programmation dans laquelle j'aimerais me spécialiser pour mon entrée sur le marché du travail dans un peu plus d'un an. J'avais donc envie de découvrir si l'infographie m'intéressait. J'ai principalement travaillé sur la mise en place de l'architecture du code, la création de plusieurs composants de l'interface utilisateur et quelques fonctionnalités comme le graphe de scène.

8.2 David Duchesne

Je suis en troisième année de génie informatique. Je suis toujours incertain par rapport à mon futur, j'aimerais faire de la programmation embarquée, mais j'ai aussi envie d'essayer la conception de jeux vidéos. C'est donc pour cette raison que je fais ce cours. J'ai travaillé sur le dessin des modèles vectoriels ainsi que sur le dessin géométrique.

8.3 Mathieu Faucher

Je suis étudiant en génie informatique 3e année et j'aimerais faire carrière en divertissement, donc en jeux vidéos ou quelque chose du genre. J'ai décidé de suivre ce cours pour faire la séquence de cours pour jeux vidéos. Lors de ce projet, j'ai travaillé sur l'histogramme, les curseurs, les imports et exports ainsi que le rapport. J'ai aussi travaillé sur les transformations et les modèles 3d.