Dungeon maker

Document de Design

***IFT-3100 Infographie* - Travail Pratique 1**

Hiver 2020

Équipe 6

BEAUDOIN, Anthony	111 153 570
CARRIER-LAMARRE, Keven	111 254 486
ST-AMAND, Alexis	111 187 985

Faculté des sciences et de génie

8 mars 2020

TABLE DES MATIÈRES

Sommaire 3

Interactivité 4

Technologie 5

Compilation 6

Architecture 8

Fonctionnalités 9

- **Exportation d'image** 9
- **Importation d'image** 11
- **Espace de couleur** 12
- **Curseur dynamique** 12
- **Outils de dessin** 13
- **Primitives vectorielles** 13
- **Formes vectorielles** 15
- **Interface** 16
- **Graphe de scène** 17
- **Sélection et modification multiple** 18
- **Transformation interactives** 20
- **Historique de transformation** 21
- **Primitives géométriques** 21
- **Modèles 3D** 22
- **Point de vue** 22

Ressources 23

Présentation 24

Annexe 25

- **Lexique:** 25

Sommaire

Le projet développé par l'équipe consiste en une application permettant à l'utilisateur de fabriquer son propre donjon. Divers objets sont mis à la disposition de ce dernier pour lui permettre de générer des salles selon son envie. Il sera donc facile pour l'utilisateur de mettre dans un espace 2D/3D s'est histoires fantastiques avec certains terrains, monstres et objets cachés qu'il pourra interactivement insérer dans la scène. Par la suite il pourra utiliser cette scène pendant les grandes soirées de Donjons et Dragons avec les copains.

L'application permet principalement à l'utilisateur d'intégrer des formes variées afin de créer une représentation graphique d'une scène ou d'une carte de donjon et dragon. Les formes 2D pré intégrées sont le pixel, la ligne, le cercle, le carré, l'ovale, le rectangle, des tuiles et une forme de "slime". Il y a également deux formes 3D qui agissent comme ces formes 2D, le cube et la sphère. Pour chacune de ces formes, il est possible prédéterminer une couleur de trait de contour, une couleur de remplissage et une épaisseur de trait. On peut également les déplacer, les faire tourner autour de l'axe des z et changer sa taille.

Il est également possible d'intégrer trois modèles 3D: Un monstre, une slime et une boule avec un visage. Ces modèle ont des caractéristiques fixes, pour l'instant.

L'utilisateur peut également importer ses propres images, en plus de pouvoir enregistrer la scène en entier et en partie.

Une caméra amovible permet à l'utilisateur d'observer sa scène sous plusieurs angles, de s'en approcher et de s'en éloigner et de déplacer la scène en même temps que les personnages.

Interactivité

L'application *Slime Dungeon* permet à l'utilisateur d'interagir de maintes façons. D'abord, l'utilisateur peut facilement tracer une certain nombre de formes paramétrables à l'intérieur de l'application. Ces formes peuvent ensuite être modifiées. La souris ainsi que de touches clavier permettent de bénéficier complètement des fonctionnalités. Parmi les paramètres disponibles, on retrouve les couleurs de remplissage et de contour, la taille des formes. Un des points forts de l'application est de pouvoir effectuer des modifications groupés. Ainsi, un groupe de formes peuvent être modifiés en une seule opération simplement en sélectionnant l'ensemble de forme désiré. Parmi les formes, il est possible de retrouver la forme *tiles* qui représente un plancher de tuiles, essentielle pour fabriquer son propre donjon. Il ne faut pas oublier non plus la forme *slime* qui peut être créée, composante cruciale. En cas d'erreur, il est possible d'effectuer des undo/redo pour annuler ou rétablir les dernières actions effectuées.

Ensuite, l'utilisateur a la possibilité d'importer une image de son cru à l'intérieur de l'application. Il suffit de glisser l'image en question dans la fenêtre ou bien d'utiliser un bouton prévu à cet effet.

Il est également possible pour l'utilisateur de prendre une capture d'écran de son travail. À tout moment, il peut appuyer sur le bouton prévu à cet effet afin d'exporter une image de la fenêtre de l'application ou bien d'une partie de cette dernière.

L'application *Slime Dungeon* permet donc à l'utilisateur de créer un grand nombre de formes et d'enregistrer le résultat sous forme d'image. Il est également possible d'inclure des images choisies par l'utilisateur, bien que la modification de celles-ci demeurent restreintes.

Technologie

openFramework a été largement utilisé à l'intérieur de l'application. L'interface graphique développé est basée sur l'utilisation de ce framework. La logique applicative repose également sur ce framework. Ce dernier intègre beaucoup d'éléments importants pour l'application telles que les images, les caméras, le traçage des formes et des paramètres associés et bien plus encore.

L'IDE Visual Studio 2017 a été utilisé afin de développer le projet. Cet environnement de développement permettait d'intégrer facilement *openFramework* avec l'aide de l'outil disponible avec *openFramework*, *openFrameworkProjectGenerator*. De plus Visual Studio permettait d'effectuer l'intégration des différentes classes facilement.

Compilation

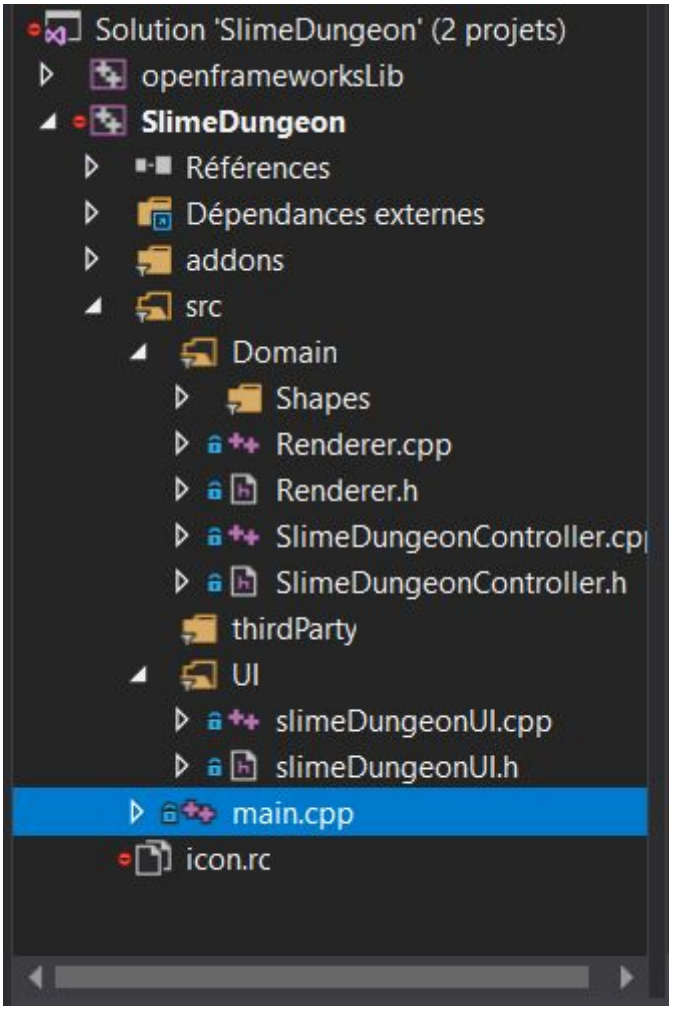
L'application *Slime Dungeon* a été bâti à l'intérieur de l'environnement de développement Visual Studio 2017 avec le framework *openFramework*. Afin de pouvoir la compiler et utiliser adéquatement l'application voici des informations à connaître:

- *openFramework* est nécessaire au fonctionnement de l'application. Il est disponible à l'adresse suivant : https://openframeworks.cc/download/.
- Les addons suivant de *openFramework* sont requis : *ofxGui*, *ofxAsimpModelLoader*. Afin de les incorporer au projet, l'application *openFrameworkProjectGenerator* a été utilisé.
- À l'intérieur du dossier src, le fichier main.cpp inclue le fichier UI/slimeDungeonUI.h. Ce dernier inclue le fichier Domain/slimeDungeonController.h. Le controleur, lui inclue Domain/Renderer.h. Ultiment, ce dernier inclue les fichiers contenu dans le dossier Domain/Shape/*
- L'affichage de certains curseurs dépend de fichier .png. Ces fichiers intitulés *fleche.png*, *hand.png* ainsi que *partialScreenshotCursor.png* doivent se retrouver à l'intérieur du dossier bin/data/ du répertoire du projet.

Afin de pouvoir compiler de nouveau dans **Visual Studio 2017**, voici la procédure à suivre :

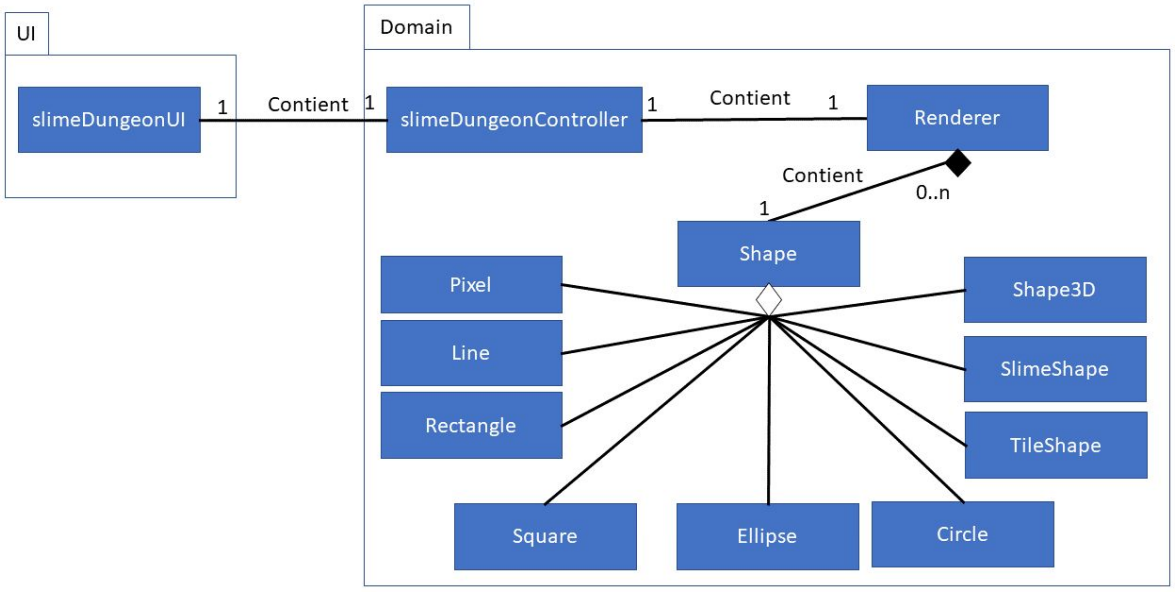
1. Obtenir *openFramework* si ce n'est pas déjà fait https://openframeworks.cc/download/
2. Ouvrir *openFrameworkProjectGenerator* . Sélectionner l'emplacement où se trouve le dossier *src* pour créer le projet. Il faut sélectionner les addons *ofxGui*, *ofxAsimpModelLoader* dans *openFrameworkProjectGenerator* .
3. Cliquer sur le bouton *Generate* afin de créer le projet. (Il est possible que le bouton soit *update* si le projet a déjà été généré.)
4. Ouvrir la solution Visual Studio ainsi générée.
5. Dans la solution, il faudra supprimer les fichiers *ofApp.h* et *ofApp.cpp*. Ces fichiers sont générés automatiquement par *openFrameworkProjectGenerator* pour contenir l'application de base. Toutefois, dans notre cas le rôle de *ofApp.h* et *ofApp.cpp* seront comblés par *slimeDungeon.h* et *slimeDungeonUI.cpp*.
6. Générer la solution.

Voici une image à titre indicatif de ce à quoi doit ressembler l'arborescence dans Visual Studio 2017 :



Architecture

L'application *Slime Dungeon* est divisé en deux paquets, un orienté vers le UI qui contient la classe *slimeDungeonUI*. Cette classe communique avec le domaine via la classe *slimeDungeonController*. Le controleur doit transmettre les informations du UI vers les autres classes pertinentes du domaine. Dans l'état actuel de l'application, le controleur n'interagit qu'avec la classe *Renderer*. Cette dernière renferme la logique applicative. L'intégralité des manipulations des formes (traçage, transformation) s'effectuent à l'intérieur de cette classe. Afin de simplifier le traçage des différentes formes, une interface générique *Shape* a été implémenté. De cette classe, l'ensemble des formes utilisées par l'application sont héritées. Ceci permet d'utiliser les concepts de polymorphismes, facilitant ainsi l'utilisation de nombreuses fonctionnalités à l'intérieur de l'application. L'architecture de l'application est illustrée dans la figure suivante:



Fonctionnalités

L'application présente un certain nombre de fonctionnalités qui sont décrites dans la présente section. Une vidéo présentant le projet est également disponible ici. (https://youtu.be/QM9lR63idpU)

Exportation d'image

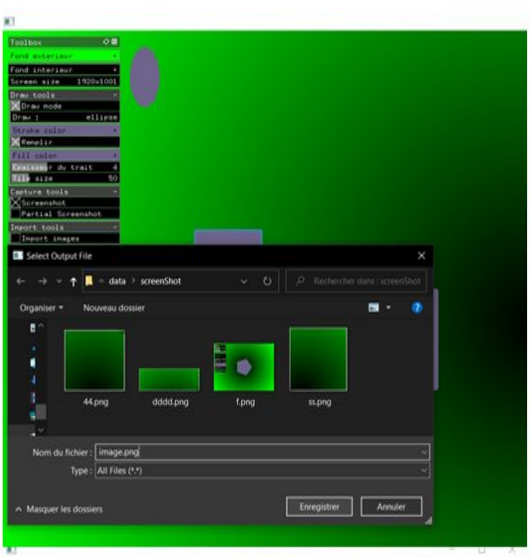
Le UI possède un onglet intitulé «Capture Tools» qui permet d'effectuer la capture de la fenêtre de l'application et d'exporter le résultat sous une image au format .png. Il y a deux manières d'utiliser cette fonctionnalitée qui font appels aux boutons «Screenshot» et «Partial screenshot».

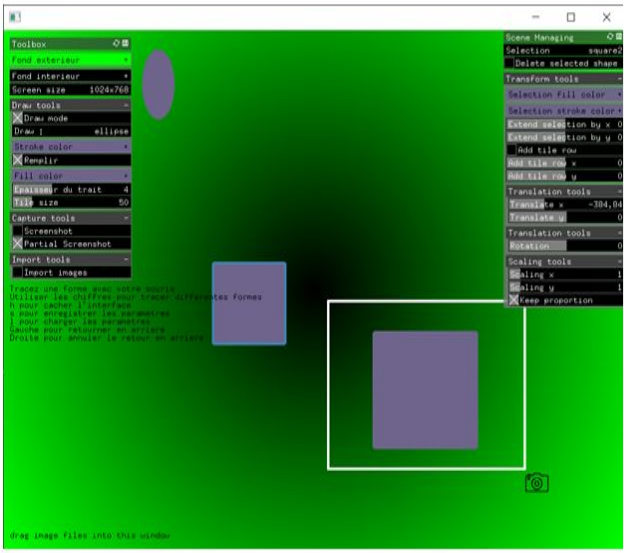
D'abord, un appui sur le bouton «Screenshot» déclenche l'exportation. Un navigateur de fichier s'ouvre et l'utilisateur peut indiquer l'emplacement où il désire sauvegarder la capture ainsi que le nom du fichier **en prenant soin d'inclure le format .png au nom**.

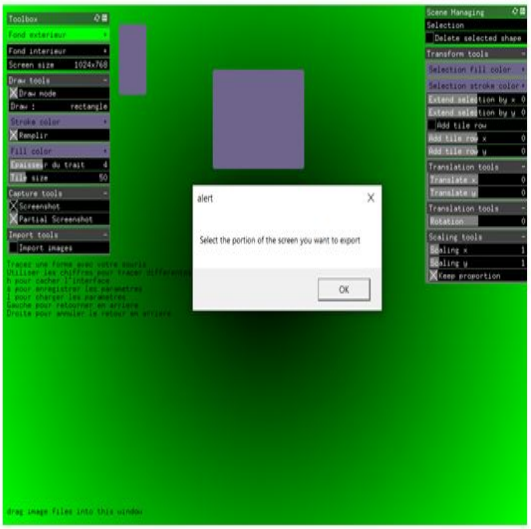
La seconde manière permet d'exporter des portions de l'écran. Le bouton «Partial screenshot» doit être coché. Si tel est le cas, lors de l'appui sur le bouton «Screenshot», une fenêtre demandera à l'utilisateur de sélectionner la portion de l'écran qu'il désire exporter. L'utilisateur appuie sur ok et peut étendre une zone de sélection avec un clic de la souris. Par la suite, un navigateur de fichier s'ouvre afin de permettre à l'utilisateur d'enregistrer la capture. Encore une fois, il est important de **préciser le format .png au nom du fichier**.

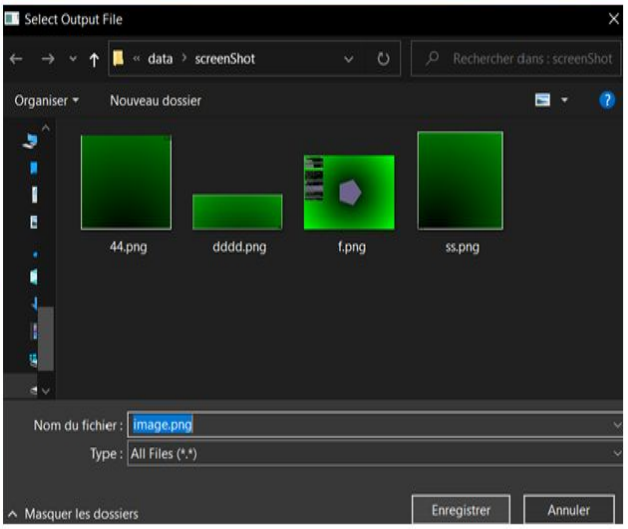
L'image dans la page suivante illustre l'exportation d'image dans l'application. On retrouve dans l'ordre de gauche à droite, de haut en bas:

1. exportation de l'écran complète. Seul le bouton «Screenshot» a été appuyé.
2. exportation partielle de l'image. Le bouton «Partial Screenshot» a été appuyé, puis le bouton «Screenshot».
3. L'avertissement qui apparaît immédiatement après avoir appuyé sur le bouton «Partial Screenshot», puis le bouton «Screenshot».
4. La fenêtre d'enregistrement de la capture d'écran. Le format est assez standard. Il ne faut pas oublier de mettre l'extension .png au nom.


Toolbox
Select Output File
data › screenShot
Rechercher dans : screenShot
Organiser
Nouveau dossier
44.png
dddd.png
f.png
ss.png
Nom du fichier : image.png
Type : All Files (*.*)
Masquer les dossiers
Enregistrer
Annuler


Toolbox
Scene Managing
drag image files into this window


alert
Select the portion of the screen you want to export
OK
drag image files into this window

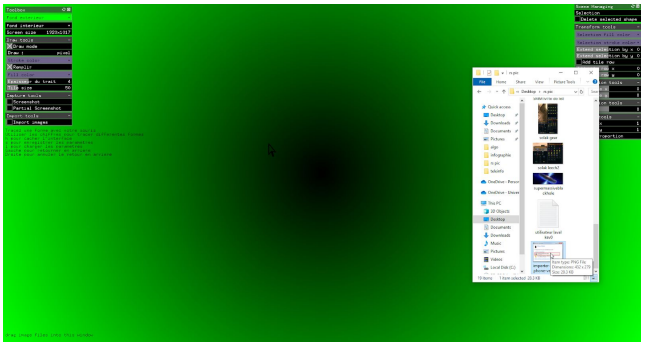

Select Output File
data › screenShot
Rechercher dans : screenShot
Organiser
Nouveau dossier
44.png
dddd.png
f.png
ss.png
Nom du fichier : image.png
Type : All Files (*.*)
Masquer les dossiers
Enregistrer
Annuler

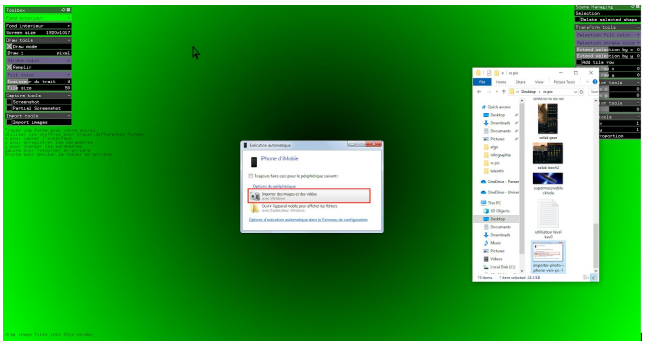
Importation d’image

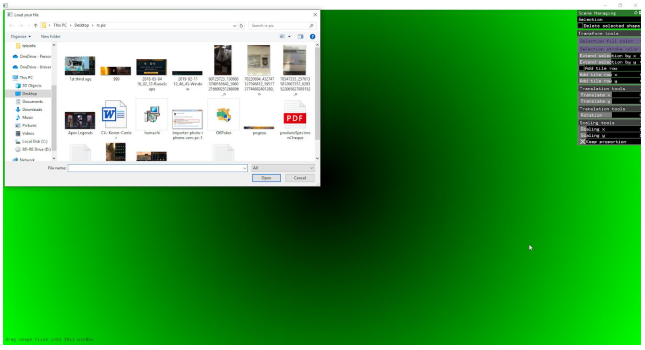
Le UI possède un onglet nommé importation d’image qui vous demande un chemin pour votre image et l’importe dans la scène. Aussi, vous pouvez tout simplement déplacer votre photo directement sur la scène pour la mettre dans celle-ci ce qui rend l'expérience beaucoup plus interactif.

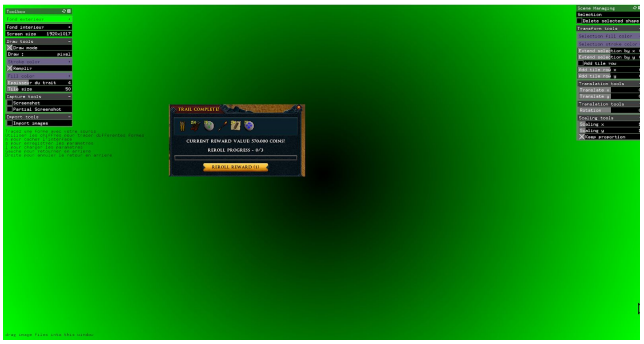
Lorsque vous appuyez sur la case import image, un navigateur de fichier apparaît pour vous demander le fichier image que vous voulez importer dans la scène. Ensuite votre image sera dessiner dans cette même scène dans un endroit choisi arbitrairement.

La façon d’importer une image de manière plus interactive est de tout simplement glissez votre image dans la scène et elle sera dessiner directement sur celle-ci. Elle sera dessinée à l’endroit ou vous avez lâché votre image.



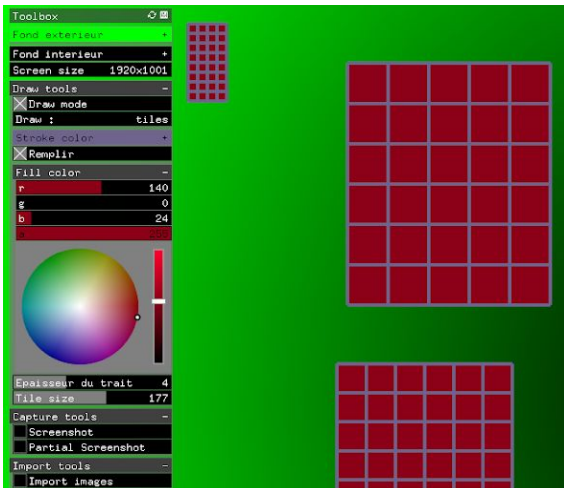






Espace de couleur

L'application permet à l'utilisateur d'utiliser des espaces de couleur RGBA et HSB afin de définir la couleur du trait ainsi que la couleur de la forme vectorielle qu'il s'apprête à dessiner. De même, il peut également changer la couleur intérieur et la couleur extérieur du fond d'écran.



Pour se faire, la couleur des traits et de l'intérieurs des formes vectorielles à dessiné et les couleur du gradient du fond d'écran dépendent de ofParameter<ofColor> intégrés au ofxGuiGroup. Une icône à droite du GUI permet d'indiquer les couleurs dont disposera la forme à dessiner.

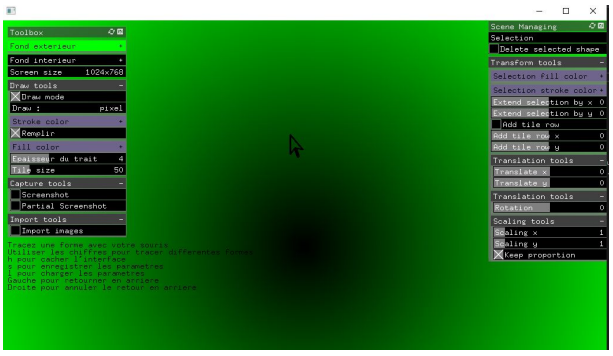
Curseur dynamique

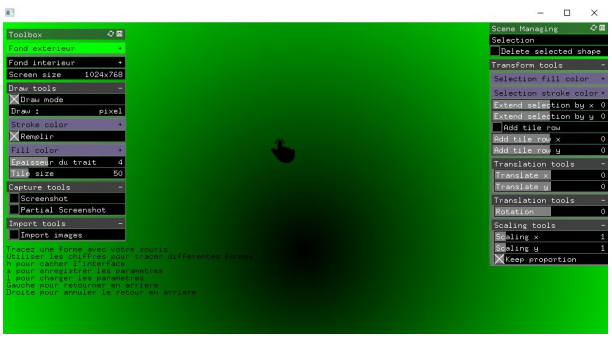
Les différents curseur de souris utiliser dans notre application change au fur et à mesur que l'utilisateur effectue différentes tâches dans l'environnement.

Il y a donc 3 différents curseurs:

- Une flèche de base lorsque l'utilisateur ne fait rien en particulier.
- Une main lorsque l'utilisateur appuie sur la souris (pour mettre des primitives par exemple).
- Une caméra lorsque l'utilisateur veut prendre une image partiellement pour la sauvegarder.

Sinon lorsque l'utilisateur met la souris au dessus d'un des GUI, la souris de base du système d'exploitation sera de retour.





Outils de dessin

L'application permet à l'utilisateur de définir la couleur du trait ainsi que la couleur de la forme vectorielle qu'il s'apprête à dessiner. De même, il peut également changer la couleur intérieur et la couleur extérieur du fond d'écran. Pour se faire, la couleur des traits et de l'intérieurs des formes vectorielles à dessiné et les couleur du gradient du fond d'écran dépendent de ofParameter<ofColor> intégrés au ofxGuiGroup. Une icône à droite du GUI permet d'indiquer les couleurs dont disposera la forme à dessiner. Grâce à des ofParameter<float> intégré au ofxGuiGroup, l'utilisateur peut définir l'épaisseur du trait ou du trait de contour de la forme avant de la dessiner.

Cela a été réalisé en liant simplement ces paramètres aux valeur de couleur ou d'épaisseur de trait de la forme à dessiner avant de l'ajouter au vector de formes à dessiner. Lorsque la forme est dessinée, ces valeurs de couleur et d'épaisseur de trait sont immuables. Le fond d'écran a en permanence les valeurs de couleur indiquées par le GUI, donc il est en tout temps possible de modifier ses couleurs.

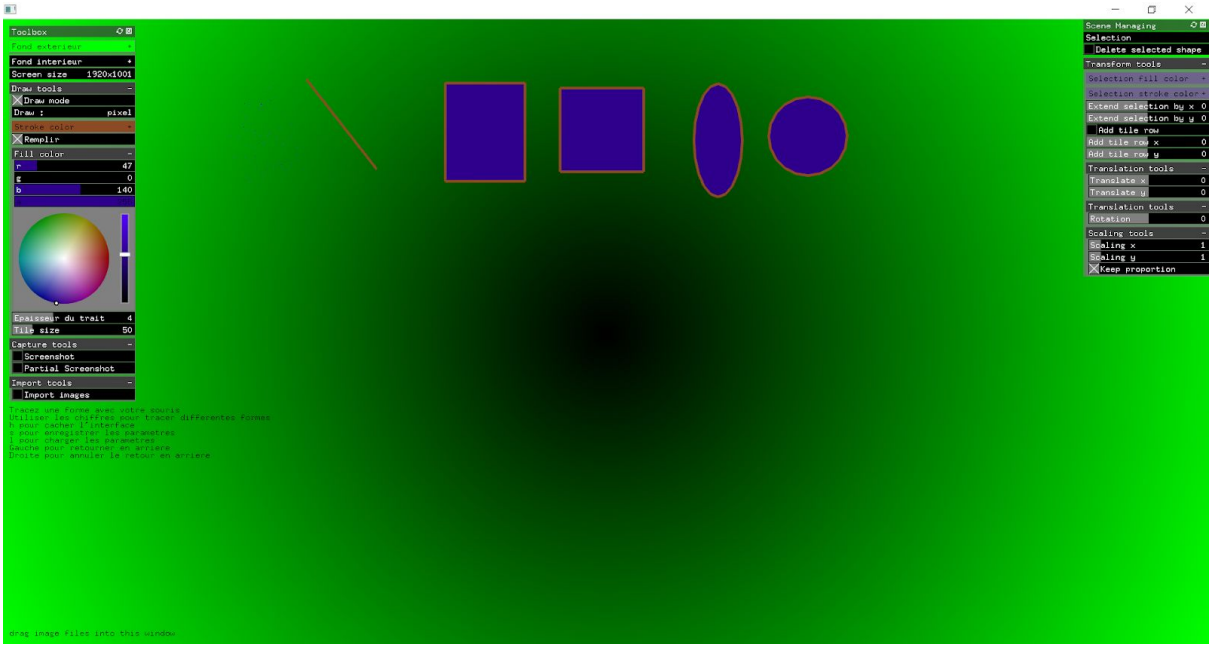
Primitives vectorielles

L'application permet de dessiner jusqu'à 6 primitives vectorielles. Ces primitives sont : le pixel, la ligne, le rectangle, le carré, l'ellipse et le cercle. Les outils associés au dessin se retrouvent dans l'onglet «Draw Tools». D'abord, le bouton «Draw mode» doit être coché afin de réaliser quelque dessin que ce soit. Par défaut, ce bouton est coché au lancement de l'application. L'utilisateur sélectionne les paramètres appropriés dans cet onglet, puis il peut dessiner en réalisant une zone de sélection avec la souris (presser + glisser + relâcher). La forme se réalise en prenant compte de la taille de la zone de sélection. Afin d'alterner entre les formes, l'utilisateur appuie sur les touches 1 à 6 du clavier. L'association touche-forme est la suivante :

Touche pressée	Forme dessinée
1	pixel
2	line
3	rectangle
4	square

5	ellipse
6	circle

Le UI dessine un échantillon du résultat à droite de la «Toolbox» et dans l'onglet «Draw Tools», le nom de la forme à être dessiné est indiqué. Ainsi, l'utilisateur peut rapidement connaître quelle primitive vectorielle sera créé et à quoi elle ressemblera. Il est bon de savoir que chaque forme générée possède un identifiant unique qui prend la forme suivante «nomForme_numero». Un exemple est plus clair dans ce cas : «circle_1» pour le premier cercle créé, «square_3» pour le troisième carré. On peut voir le résultat de chaque primitive vectoriel dans la figure suivante (une série de pixel a été dessiné afin de pouvoir observer le résultat):



Il est à noter que le carré et le cercle sont des primitives régulières. Lorsqu'elles sont actives et que «Draw mode» est coché, la zone de sélection prendra une forme carré plutôt que de suivre exactement le curseur. Ceci permet de générer une forme conforme à la zone de sélection.

Formes vectorielles

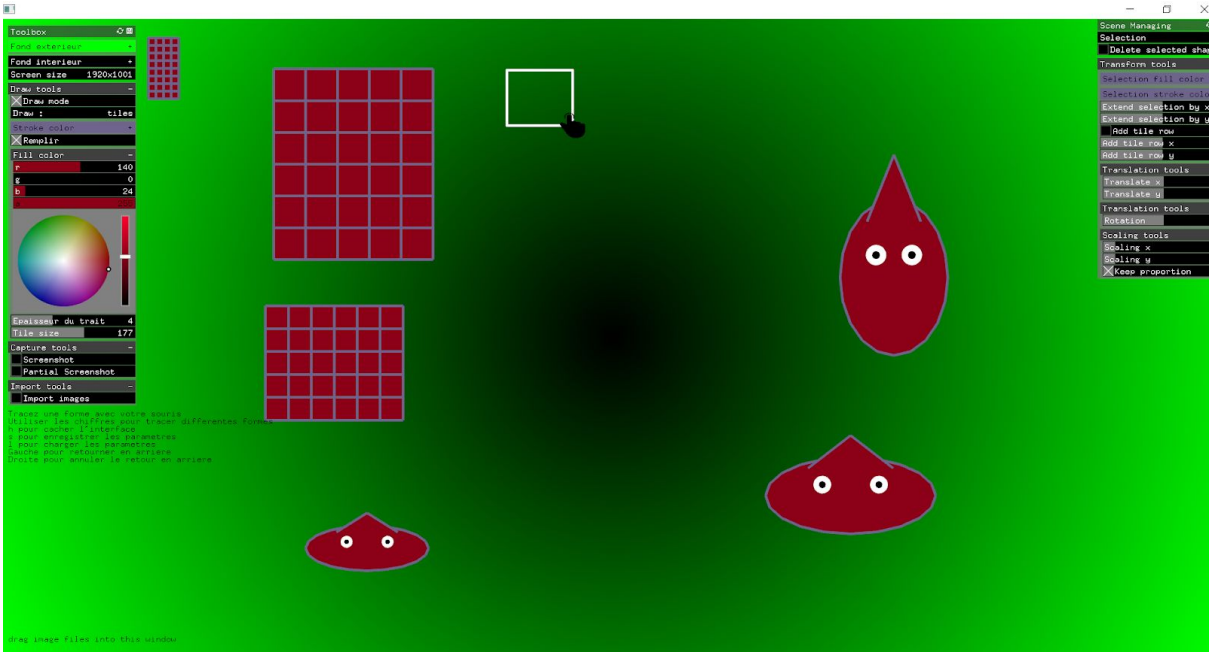
De manière similaire aux primitives vectorielles, il est possible de dessiner certaines formes prédéfinies dans l'application. Deux de ces formes sont disponibles, la forme «slime» et la forme «tiles». Afin d'activer le dessin de ces formes, l'utilisateur presse sur certaine touche qui sont les suivantes:

Touche pressée	Forme dessinée
7	tiles
8	slime

La forme tiles correspond à une série de tuiles disposés côte-à-côte. Lors de la création de cette forme, la zone de sélection est fragmentée selon le nombre maximal de tuile entière qui peut y entrer. Par exemple, si l'utilisateur génère une zone de sélection de 202 pixel de large alors que les tuiles mesurent 50, il n'y aura que 4 tuiles. L'utilisateur peut régler la taille des tuiles à son aise. Ceci dit, si une zone de sélection de taille trop petite est dessinée (soit que le motif ne comporterait aucune tuile) aucune forme n'est créée.

La forme slime fonctionne de manière parfaitement analogue aux autre primitives vectorielles. Il n'y a donc rien de particulier à mentionner à son sujet, outre qu'elle est primordiale lors de la création d'un slime dungeon.

L'image suivante permet de mieux voir le résultat du traçage des formes vectorielles en question.



On peut d'abord noter sur la figure que quelques slimes ont été dessinés avec des zones de tailles différentes. Ensuite, on constate que deux formes *tiles* ont été créées en utilisant une taille de tuiles différentes. Finalement, la zone de sélection qu'il est possible de voir sur l'image illustre une zone de taille invalide. La taille des tuiles a été réglé à une valeur de 177. Ceci signifie qu'avec la sélection, aucune tuile ne peut être placée. Ceci ne mènera donc pas à la création d'une forme tuile.

Interface

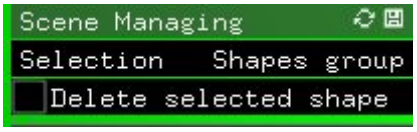
Un texte sous le panneaux de contrôle gauche indique à l'utilisateur les information nécessaires pour utiliser l'application. Ce texte à été généré grâce à des ofDrawBitmapString et un vecteur de string qui contient chaque instructions.

Graphe de scène

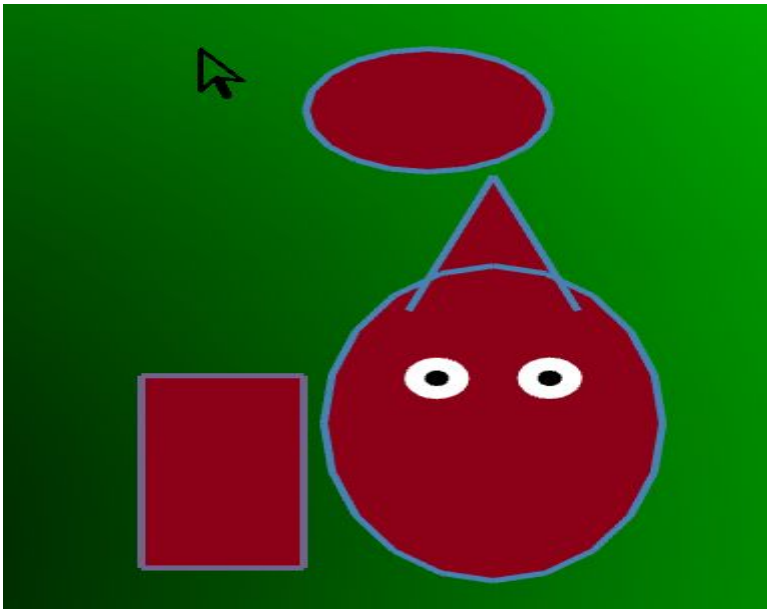
L'application offre la possibilité de manipuler les formes créées. Afin de permettre cela, il faut d'abord s'assurer d'avoir désactivé «Draw mode» de l'onglet «Draw Tools». Une fois cela fait, l'utilisateur peut cliquer sur les formes qu'il désire sélectionner. Ces dernières apparaîtront alors avec un contour en surbrillance. De plus, dans le panneau «Scene» située par défaut à droite de la fenêtre, l'utilisateur peut consulter le nom de la figure sélectionnée. Advenant une sélection de plusieurs éléments, il y est indiqué «Shapes group». Pour effacer la sélection, l'utilisateur peut soit cliquer sur l'élément qu'il désire retirer ou bien cliquer sur aucune surface pour tout désélectionner.

L'utilisateur peut également remarquer le bouton «Delete selected shape». Ce bouton fait exactement comme son nom l'indique: il supprime les formes sélectionnées.

Les deux images suivantes indique la récupération des noms de la sélection active.



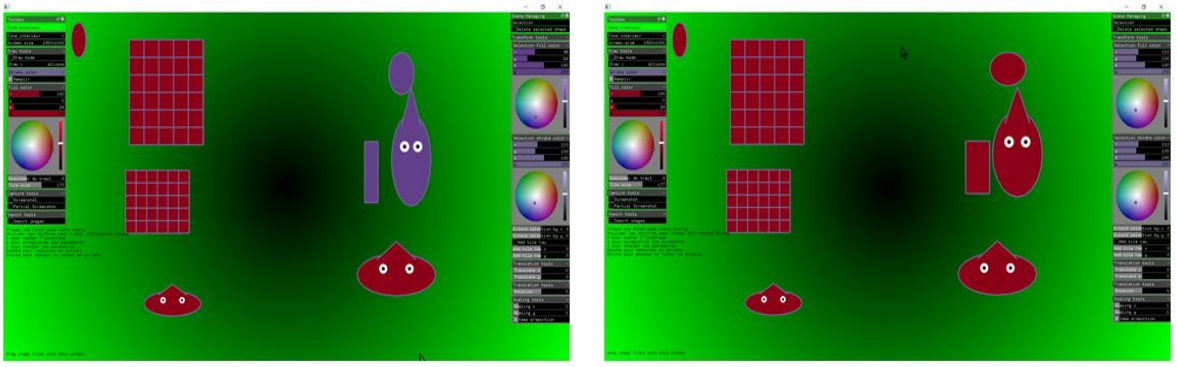
L'image qui suit illustre deux formes sélectionnées (ellipse et slime) ainsi qu'une forme qui ne l'est pas.



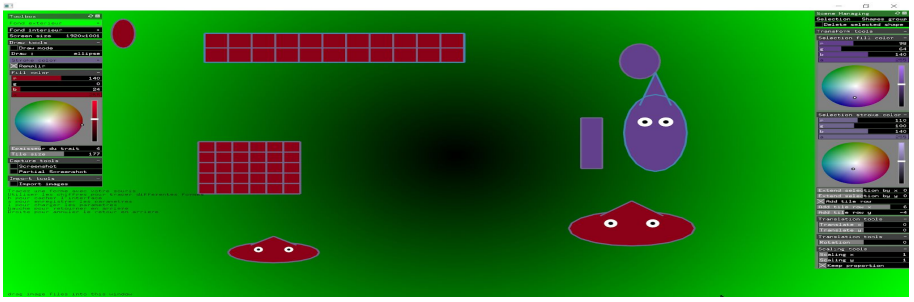
Sélection et modification multiple

Les formes dessinées par l'utilisateur possède un certain nombre d'attributs en commun. Il est donc possible de modifier certains attributs d'une sélection de forme en une seule action. (Note : tout ce qui est décrit dans cette section peut évidemment être exécutée sur une sélection d'une seule forme.)

Dans l'onglet «Transform Tools» du panneau «Scene», l'utilisateur peut modifier la couleur de contour et la couleur de remplissage. Il peut également étendre la bordure de droite ou la bordure inférieure qui délimite la forme (*bounding box*) en utilisant «Extend selection by». Il est impossible d'étendre la bordure à une distance inférieur du point d'ancrage supérieur droit. En d'autre termes, la forme ne peut pas être renversée sur elle-même. La figure suivante illustre certaines modification qui peuvent être effectuées:



Dans le cas des tuiles, il est possible d'ajouter des rangées ou colonnes facilement en cochant l'option «Add tile row», puis en manipulant les paramètres correspondant. Advenant une sélection qui comporte des formes qui ne sont pas des tuiles, ces modifications ne seront pas appliquées. La figure suivante illustre ceci (la tuile modifiée est celle qui se retrouve au même endroit dans l'image plus haut):



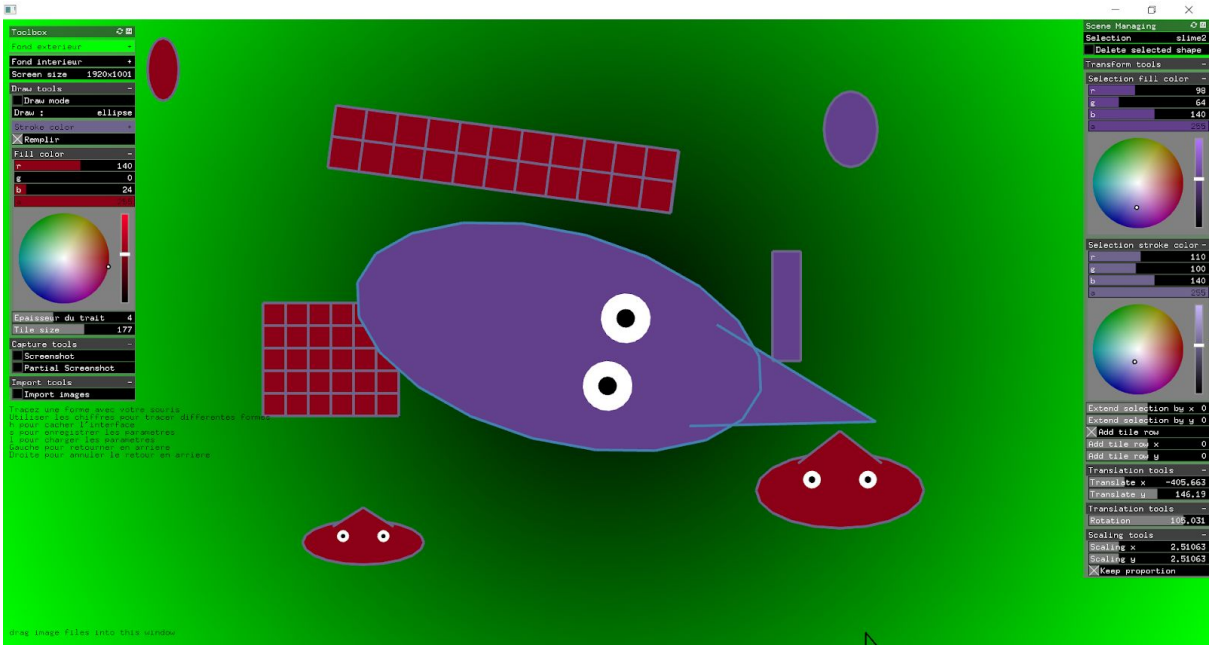
Note: «Add tile row» doit être désactivé afin de permettre d'étendre la *bounding box* des formes de la sélection.

Les modifications de la *bounding box* demeurent actives jusqu'à ce que l'utilisateur désélectionne tous les éléments sélectionnés.

Transformation interactives

L'application permet d'effectuer des translations, des rotations ainsi qu'un ajustement de l'échelle sur les formes. Ces outils se retrouvent tous dans l'onglet «Transform Tools» du panneau «Scene». Le fonctionnement est similaire à la modification d'attribut des formes, à savoir que les transformations effectuées s'exécutent sur l'ensemble des formes sélectionnées.

La translation permet de déplacer les formes selon x ou y. La rotation fait pivoter les formes selon l'axe des z et sur elle-même. La mise à l'échelle augmente la proportion en x ou en y ou bien les deux en simultanées si l'option «Keep proportion» est cochée. La rotation et la mise à l'échelle s'effectuent selon le centre de la figure. La figure suivante illustre plusieurs des transformations qui ont été effectué à partir de la même figure que l'on retrouve dans la Section Sélection et modification multiples:



On peut ainsi observer qu'une forme slime a été pivotée, a subi une translation horizontale et verticale et a également été mis à l'échelle. De plus une forme tile a été pivotée.

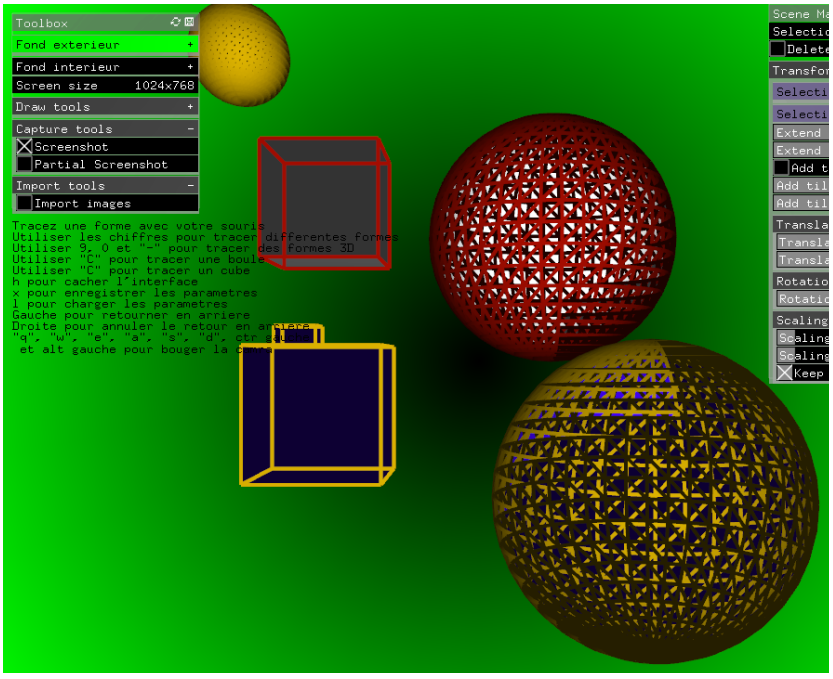
Historique de transformation

En cas d'erreur, l'utilisateur peut utiliser la flèche de gauche pour annuler ses derniers dessins et retourner en arrière. S'il retourne trop dans le passé, il peut alors revenir grâce à la flèche droite. (Gauche: *undo*, Droite: *redo*). Cependant, s'il applique une modification au nombre de formes vectorielles présentes en en dessinant ou effaçant, il perdra trace de l'historique "futur" et ne pourra que retourné en arrière.

Ce critère a pu être possible grâce à la structure de données utilisée pour dessiner les formes. Puisqu'elles sont stocké dans un vecteur et tracer en ordre, on a qu'à retirer le dernier élément de ce vecteur pour effectuer un retour en arrière. Pour pouvoir refaire ce que l'utilisateur a annulé, on a simplement empiler sur une pile les éléments au fur et à mesure qu'on les retirait du vecteur. Ainsi, pour refaire, il s'agissait de dépiler et de placer la forme à la fin du vecteur.

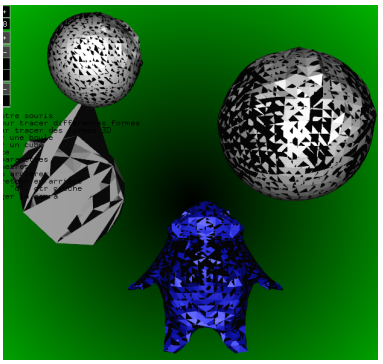
Primitives géométriques

Au même titre que l'utilisateur peut dessiner les 6 primitives vectorielles différentes, il peut également dessiner un cube et une sphère. Ces derniers agissent exactement comme les autres primitives, mais ont un axe de profondeur centré sur le plan xy. C'est à dire que leur centre est en (x, y, 0). L'utilisateur peut également modifier la couleur de remplissage et de trait ainsi que l'épaisseur des lignes de contour. Pour le cube, ces lignes correspondent à ses arrêtes, pour la sphère, à son armature en fil de fer. Du côté du code, c'est le même que pour les carrés et les cercle, mais avec les fonctions ofDrawBox et ofDrawSphere.



Modèles 3D

Pour ce critère, nous avons conçus grâce à Blender 3 modèles 3D. Un représente un monstre imaginaire, un autre est une représentation 3D de notre “slime” et le dernier est une balle avec un visage. Malheureusement, nous avons dû réduire le nombre de faces de nos modèles afin d’éviter un énorme lag.



Nous avons ensuite simplement traiter ces modèles comme des Shape en ajustant les fonctions de dessin pour des modèles 3D et avons ajouté une lumière géré par la couche application pour ne pas que le rendu soit tout noir lorsqu’on dessine des modèles. Nous ne pouvons pas non plus changer leur couleur ou les sélectionner, mais ils fonctionnent avec annuler et refaire (les flèches).

Point de vue

L’utilisateur peut modifier la caméra de différente façons. Avec contrôle et alt gauche il peut avancer et reculer la caméra (dolly). Avec q et e, il peut la faire tourner en sens antihoraire et en sens horaire. Les touches w, a, s, d permettent de la faire glisser sur les axes x et z.

Du côté programmation, lorsqu’on appuis sur une touche, le paramètre correspondant est mis à true et active ainsi l’effet de caméra. Lorsque la touche est relâché, le paramètre est mis à false et l’effet de caméra cesse de s'exécuter.

Ressources

Ressources originale:

- ball.dae
- monster.dae
- slime.obj
- slime1.dae

(ces modèles 3D ont été produit sur blender)

Références:

- Exemples du professeur
- Documentation open frameworks: https://openframeworks.cc/documentation/

Présentation

Anthony Beaudoin est un étudiant de deuxième année au baccalauréat en informatique. Grand curieux, il a toujours aimé développé ses connaissances dans de multiples domaines. Après plusieurs années d'études dans le domaine agricole, il décide d'effectuer un changement de cap afin de retrouver une autre de ses grandes passions : le domaine informatique. Réalisateur, écrivain et maintenant informaticien, il continue d'endosser de nombreux chapeaux alors qu'il continue de découvrir de nouvelles passions.

Keven Carrier étudie pour sa 4ième session dans le BAC en informatique. Sa passion pour le jeu vidéo et la programmation pousse ce vieillissant jeune de coeur à faire un retour aux études pour réaliser ce qu'il croyait impossible il y a 5 ans, un diplôme universitaire dans son domaine de rêve. Maintenant qu'il est si près du but, il continue de persévérer et de rêver à ce jour ou il aura son emplois tant voulu.

Alexis St-Amand est un étudiant en deuxième année au baccalauréat intégré en informatique et mathématiques. Amateur d'art et de science, il a toujours aimé joindre les deux grâces à ces deux domaines d'études et particulièrement dans le domaine du jeu vidéo ou de l'informatique appliqué aux sciences.

Annexe

Lexique:

UI: user interface. Désigne l’interface visible par l’utilisateur

bounding box: Boîte minimale qui peut contenir une forme

GUI: interface utilisateur graphique