David Gaudet (111 222 237)

Didier Michaud (536 955 604)

Laurie Théberge (536 950 573)

Patrick Veilleux (536 907 694)

Document design – Infographie

Présenté à :

Philippe Voyer

Dans le cadre du cours :

IFT-3100

Le 05 mars 2022

Université Laval

Sommaire

Pour ce projet, nous avons pris la décision de procéder au développement d'un « dungeon creator », ayant pour but la création et l'édition de cartes de donjon. Un tel logiciel pourrait s'adresser autant à un « dungeon master » lors d'une campagne de jeu de rôle qu'à un créateur de contenu en ligne.

La vue orthogonale permet à l'utilisateur d'avoir une vue d'ensemble de sa carte tout en la modifiant aisément. Il lui est possible d'ajouter des murs délimitant les corridors du donjon et d'ajouter des sphères, coffres, torches et ennemis. Il est également possible d'ajouter des formes géométriques en deux dimensions pouvant servir de marqueur.

La vue en perspective, quant à elle, permet d'avoir un aperçu des primitives, des modèles, de l'éclairage et des textures dans un espace en trois dimensions. Le but est de complémenter la vue orthogonale lors de la création et l'édition en offrant un aperçu à la première personne des éléments constituant le donjon.

Cette idée nous est venue, à l'origine, car chaque membre de l'équipe est un amateur de jeux vidéo et de jeux de rôle. Nous savions qu'avec un projet nous tenant ainsi à cœur, nous serions prêts à investir le temps et les efforts pour livrer un travail de qualité.

Interactivité

Plusieurs comportements interactifs sont possibles dans l'application. Si nous souhaitions qu'il soit facilement utilisable par un usager désirant construire lui-même un donjon ou une taverne d'aventuriers, il était nécessaire que les possibles interactions soient les plus instinctives possibles.

Premièrement, au niveau des caméras, nous avons la caméra de perspective qui peut être déplacée avec la souris dans le but de permettre à l'utilisateur de tourner autour d'une origine. L'origine de la caméra du haut, qui en en vue orthogonale, peut être déplacée grâce aux flèches du clavier alors que celle de la caméra du bas, qui est en vue en perspective peut l'être grâce aux boutons de la souris.

Il est possible de choisir, dans un menu de l'interface graphique, les objets que nous souhaitons dessiner dans la grille en prochain. Pour les formes en trois dimensions, nous avons le choix entre un cube (représentant les murs du donjon), une sphère, un coffre, une torche sur pied et un vilain Armadillo. Pour dessiner cesdites formes, suite à la sélection, il suffit de cliquer dans la grille de la caméra en vue orthogonale (qui est en haut dans le logiciel) pour que l'objet apparaisse dans la case choisie.

Nous pouvons également choisir la couleur des objets qui se trouvent dans la scène avec l'interface graphique "Material Editor". En cliquant sur un objet dans la grille en vue orthogonale, il est possible d'ensuite choisir sa couleur ambiante (ambient), sa brillance (shininess), sa couleur diffuse ainsi que la "specular color" et "emissive color".

Nous avons également ajouté un moyen de changer la grosseur des cases de la grille ainsi que la quantité de cases dans la grille. Ceci nous permet d'avoir une surface d'une taille personnalisée pour créer le donjon souhaité.

Pour terminer, nous avons ajouté l'option de pouvoir enregistrer les objets ajoutés avec leurs modifications avec une fonction « save ». Nous pouvons aussi afficher l'ensemble des objets sauvegardés au préalable avec une fonction « load ». Il est également possible d'utiliser les fonctions « undo » pour annuler une modification et « redo » pour restaurer une modification.

Technologies

Pour ce projet, nous avons pris la décision d'utiliser Visual Studio 2022 en raison de sa compatibilité avec « openFrameworks » ainsi que sa documentation abondante.

Notre choix s'est donc arrêté sur « openFrameworks », puisqu'il s'agit d'un « framework » complet, agrémenté de nombreux exemple de code. Le fait qu'il s'agisse de celui utilisé dans le cours a également joué sur notre décision.

Du côté des « addons », nous avons en premier lieu décidé d'utiliser ofxAssimpModelLoader pour tout ce qui touche le téléversement de modèles 3D.

Pour ce qui est de l'interface, nous avons orienté notre choix sur ofxImGui, qui offre un large éventail de fonctionnalités pour la création d'interfaces graphiques. Cela nous a permis d'adapter le « GUI » à nos besoins en sélectionnant les éléments d'interface les plus appropriés à notre application.

Nous avons également fait usage du logiciel Photoshop, particulièrement pour l'édition des curseurs, dans le but de s'assurer du nettoyage des modèles et d'avoir une taille uniforme.

Pour combler le manque de documentation de certaines technologies utilisées, nous avons aussi utilisé ChatGPT, bien qu'une attention particulière devait être donnée pour s'assurer d'avoir accès à de l'information valide et cohérente avec les versions des technologies utilisées.

Finalement, nous avons également utilisé le logiciel Blender dans l'édition des modèles 3D, principalement pour le mapping des textures, mais également pour recentrer les modèles, et pour nous assister dans la compréhension d'un problème de rotation de l'objet, décrit plus bas dans la section fonctionnalité (6.4.3).

Compilation

Le projet a été réalisé sur Visual Studio 2022, sous Windows.

La procédure suivante est donc optimisée pour ce système d'exploitation, mais les étapes sont sensiblement les mêmes pour OSX, avec l'utilisation de Xcode.

Il est à noter que la procédure abrégée se trouve dans le « Read me » du projet.

1- Cloner le dépôt GitHub suivant sur le poste de travail :
   git clone https://github.com/patrix87/Dungeon-Creator.git

2-

a. Option #1
   cloner le dépôt GitHub « openFrameworks », version 0.11.2 au même niveau que le dossier « Dungeon-Creator ».
   git clone https://github.com/openframeworks/openFrameworks.git --branch 0.11.2

b. Option #2
   Télécharger la version de « openFrameworks » adaptée au système d'exploitation à partir de la section « download » du site officiel (sélectionner la version 0.11.2) :
   https://openframeworks.cc/download/

3- Extraire le dossier au même niveau que Dungeon-Creator et le renommer « openFrameworks »
   Les deux dossiers doivent se trouver au même niveau

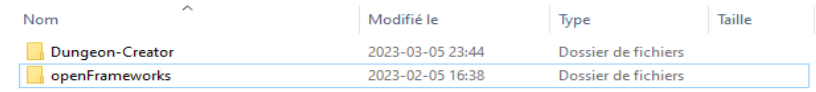
4- Cloner le « addon » OFxImGui dans le dossier « Addon » de « openFrameworks » (il sera nommé OFxImGui-Master

git clone https://github.com/jvcleave/ofxImGui.git ./openframeworks/addons/ofxImGui

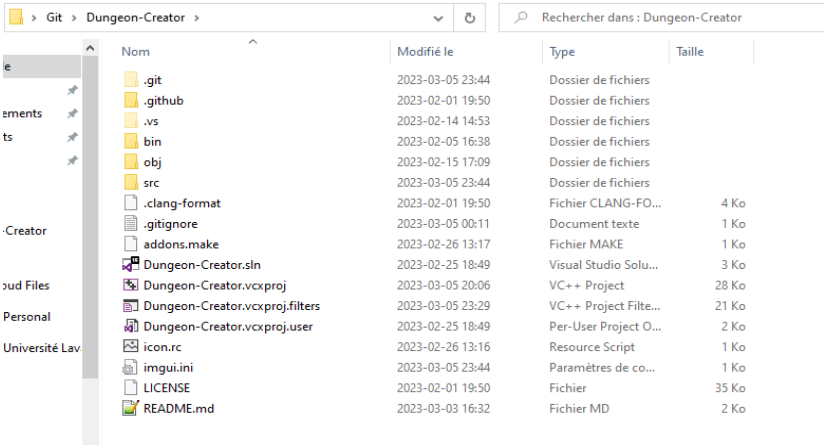
Il est important de le renommer en enlevant le « -Master » (OFxImGui)

Toujours à l'aide de git, télécharger les librairies de « openFrameworks » :

.\openFrameworks\scripts\vs\download_libs.ps1

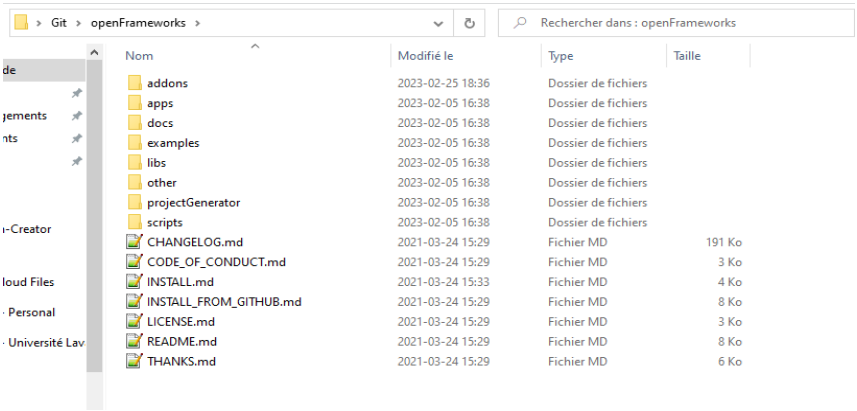


Nom	Modifié le	Type	Taille
Dungeon-Creator	2023-03-05 23:44	Dossier de fichiers	
openFrameworks	2023-02-05 16:38	Dossier de fichiers	



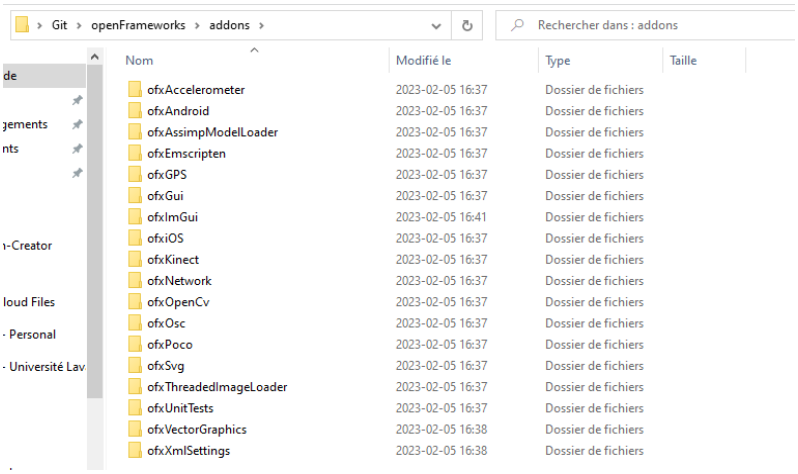
Git › Dungeon-Creator › — Rechercher dans : Dungeon-Creator

Nom	Modifié le	Type	Taille
.git	2023-03-05 23:44	Dossier de fichiers	
.github	2023-02-01 19:50	Dossier de fichiers	
.vs	2023-02-14 14:53	Dossier de fichiers	
bin	2023-02-05 16:38	Dossier de fichiers	
obj	2023-02-15 17:09	Dossier de fichiers	
src	2023-03-05 23:44	Dossier de fichiers	
.clang-format	2023-02-01 19:50	Fichier CLANG-FO...	4 Ko
.gitignore	2023-03-05 00:11	Document texte	1 Ko
addons.make	2023-02-26 13:17	Fichier MAKE	1 Ko
Dungeon-Creator.sln	2023-02-25 18:49	Visual Studio Solu...	3 Ko
Dungeon-Creator.vcxproj	2023-03-05 20:06	VC++ Project	28 Ko
Dungeon-Creator.vcxproj.filters	2023-03-05 23:29	VC++ Project Filte...	21 Ko
Dungeon-Creator.vcxproj.user	2023-02-25 18:49	Per-User Project O...	2 Ko
icon.rc	2023-02-26 13:16	Resource Script	1 Ko
imgui.ini	2023-03-05 23:44	Paramètres de co...	1 Ko
LICENSE	2023-02-01 19:50	Fichier	35 Ko
README.md	2023-03-03 16:32	Fichier MD	2 Ko



Git › openFrameworks › — Rechercher dans : openFrameworks

Nom	Modifié le	Type	Taille
addons	2023-02-25 18:36	Dossier de fichiers	
apps	2023-02-05 16:38	Dossier de fichiers	
docs	2023-02-05 16:38	Dossier de fichiers	
examples	2023-02-05 16:38	Dossier de fichiers	
libs	2023-02-05 16:38	Dossier de fichiers	
other	2023-02-05 16:38	Dossier de fichiers	
projectGenerator	2023-02-05 16:38	Dossier de fichiers	
scripts	2023-02-05 16:38	Dossier de fichiers	
CHANGELOG.md	2021-03-24 15:29	Fichier MD	191 Ko
CODE_OF_CONDUCT.md	2021-03-24 15:29	Fichier MD	3 Ko
INSTALL.md	2021-03-24 15:33	Fichier MD	4 Ko
INSTALL_FROM_GITHUB.md	2021-03-24 15:29	Fichier MD	8 Ko
LICENSE.md	2021-03-24 15:29	Fichier MD	3 Ko
README.md	2021-03-24 15:29	Fichier MD	8 Ko
THANKS.md	2021-03-24 15:29	Fichier MD	6 Ko



Git › openFrameworks › addons › — Rechercher dans : addons

Nom	Modifié le	Type	Taille
ofxAccelerometer	2023-02-05 16:37	Dossier de fichiers	
ofxAndroid	2023-02-05 16:37	Dossier de fichiers	
ofxAssimpModelLoader	2023-02-05 16:37	Dossier de fichiers	
ofxEmscripten	2023-02-05 16:37	Dossier de fichiers	
ofxGPS	2023-02-05 16:37	Dossier de fichiers	
ofxGui	2023-02-05 16:37	Dossier de fichiers	
ofxImGui	2023-02-05 16:41	Dossier de fichiers	
ofxiOS	2023-02-05 16:37	Dossier de fichiers	
ofxKinect	2023-02-05 16:37	Dossier de fichiers	
ofxNetwork	2023-02-05 16:37	Dossier de fichiers	
ofxOpenCv	2023-02-05 16:37	Dossier de fichiers	
ofxOsc	2023-02-05 16:37	Dossier de fichiers	
ofxPoco	2023-02-05 16:37	Dossier de fichiers	
ofxSvg	2023-02-05 16:37	Dossier de fichiers	
ofxThreadedImageLoader	2023-02-05 16:37	Dossier de fichiers	
ofxUnitTests	2023-02-05 16:37	Dossier de fichiers	
ofxVectorGraphics	2023-02-05 16:38	Dossier de fichiers	
ofxXmlSettings	2023-02-05 16:38	Dossier de fichiers	

Architecture TP1

Note : toutes les classes inclut ofMain

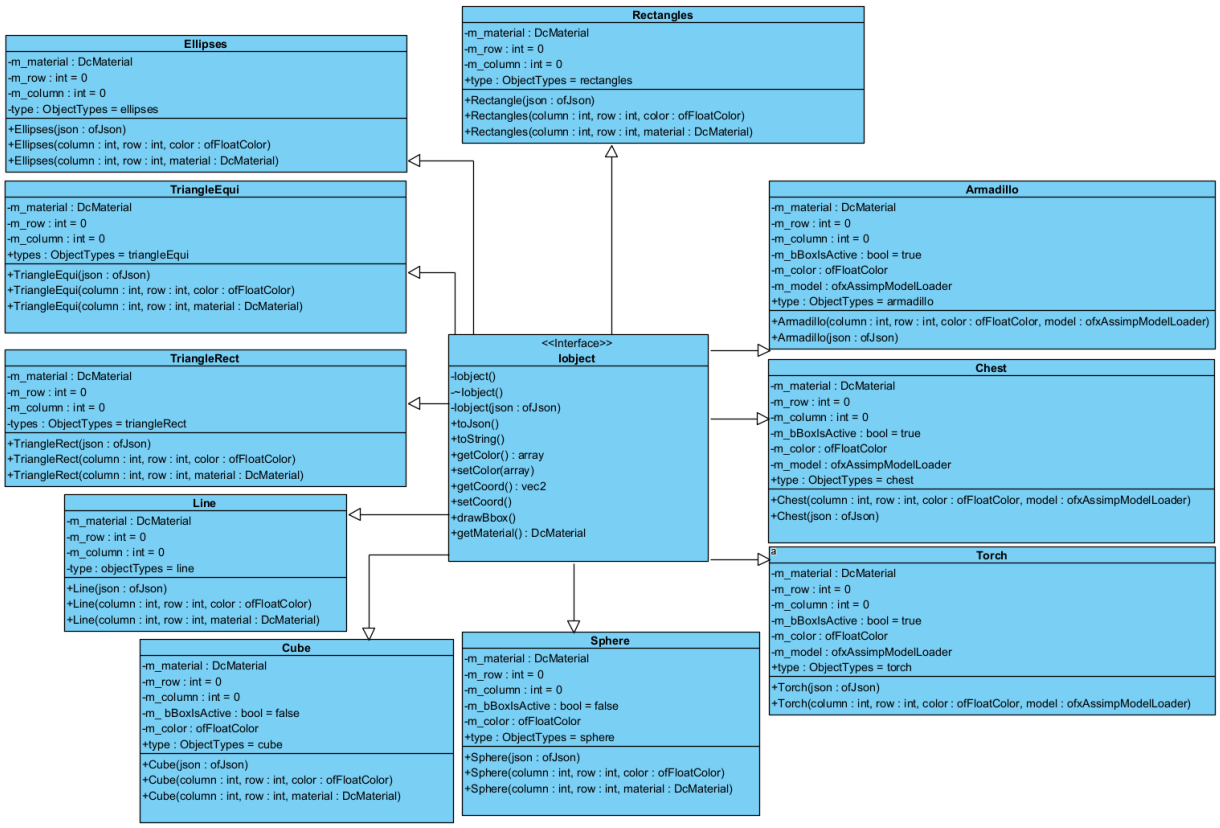
Application permet de faire le pont entre l'état de la souris (classe Cursor) et les fonctions de dessins (Scene Data et les différents objets). L'état de la souris est modifié grâce à l'interface graphique.

- include: Cameras, Cursor, Grid, Lights, SceneData, UserInterface, Viewports, ofxAssimpModelLoader

Scene Data permet de faire la gestion des différents objets et de sauvegarder l'historique de transformation dans un fichier Json.

- include:
  - algorithme, iostream, string, tuple, vector
  - Toutes les primitives (modèles 3D et 2D)

Architecture des objets : Permets d'instancier les différents éléments dessinés dans la scène.



UserInterface permet de changer le mode de souris, d'applique les modifications aux différents éléments de la scène (couleur, matériaux), de modifier la grille et de sélectionner les différentes primitives à ajouter à l'aide de boutons qui, ultimement, changent l'état de la souris.

- Include Cursor, SceneData, Viewports

Grid permet de dessiner la grille et de faire la gestion de la position des différents éléments s'y trouvant.

- Include : Cameras, Cursor et IObject

Architecture TP2

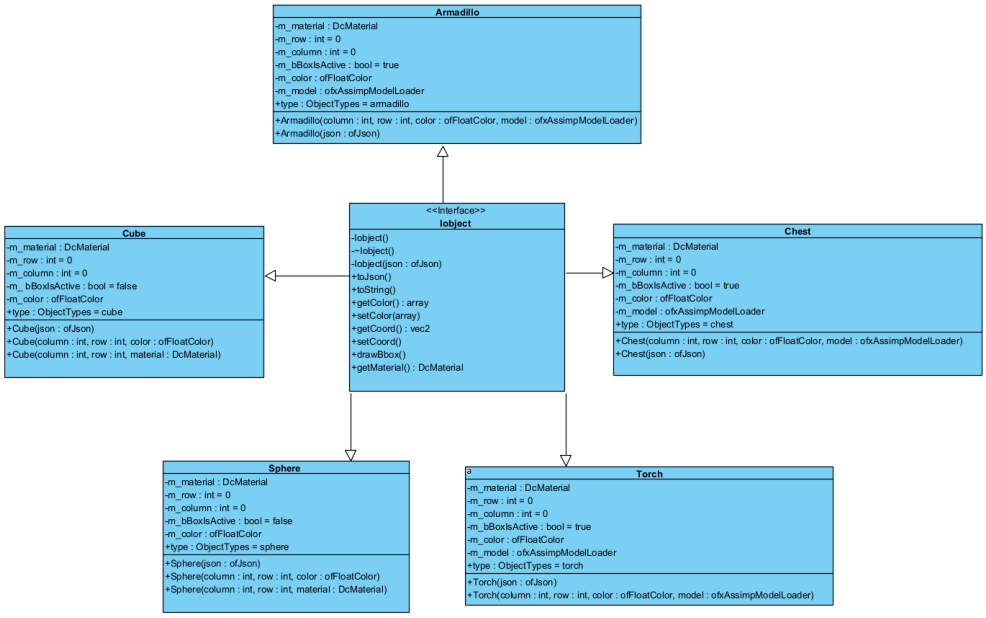
Nouvelles classes :

Shader : Classe permettant de créer un shader PBR. On y fait le « setup » et « l'update », passant au shader toutes les valeurs nécessaires à son fonctionnement.

Renderer : s'occupe de faire les appels de dessin et d'initialiser le shader.

- Include : Cameras, Courbe, Cursor, Generator, Grid, Lights, Models, Ray, Renderer, SceneData, Shader, UserInterface, Viewports

Diagramme des objets à jour :



Changements apportés aux fonctionnalités du premier livrable :

- Amélioration de la classe IObject pour réduire la répétition de code
- Amélioration de la caméra pour avoir des contrôles qui se rapprochent d'un jeu vidéo (W, A, S, D, clic gauche pour regarder autour)
- Nous avons retiré les primitives 2D

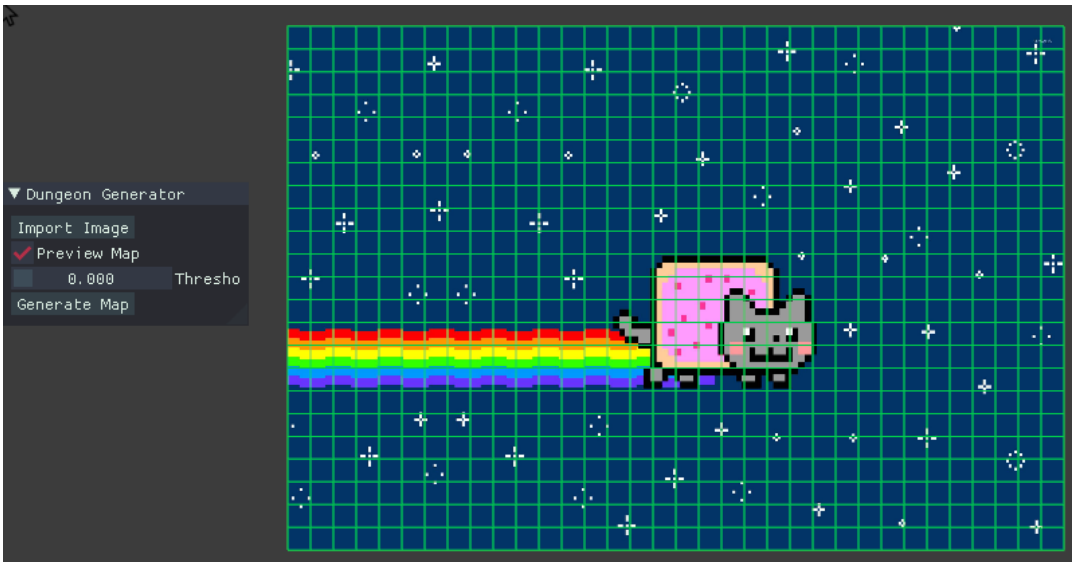
Fonctionnalités TP1

Image

1.1 Importation d'images

Il est possible d'importer des images dans l'application via le bouton `Import image` du menu « Dungeon-Creator ».

L'image importée est alors affichée en arrière-plan de la grille lors de la configuration du créateur de donjon automatique.



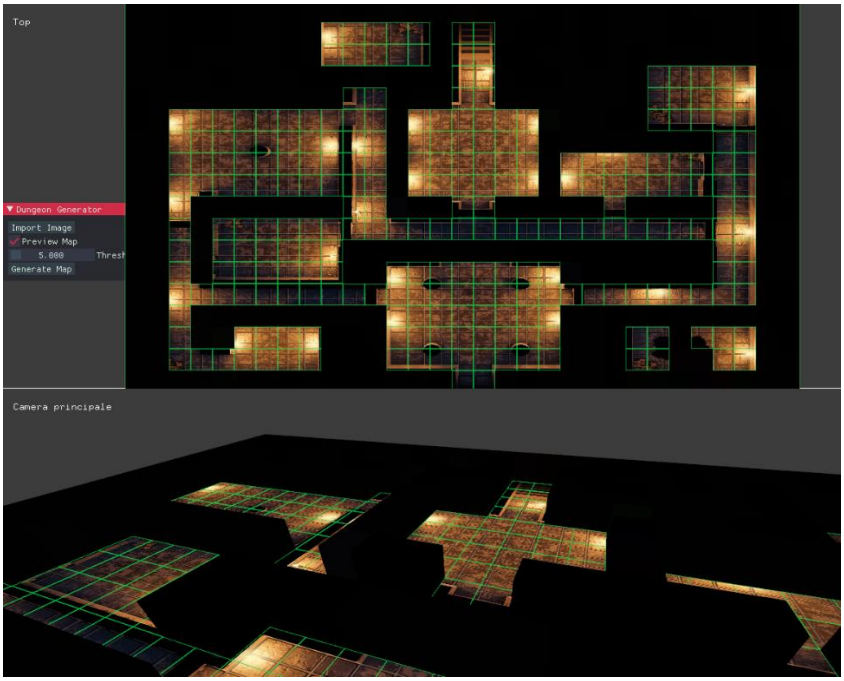
L'image est ajustée afin de prendre les dimensions de la grille.

Plus de détails sur la création automatique dans la section Bonus.

1.3 Échantillonnage d'images

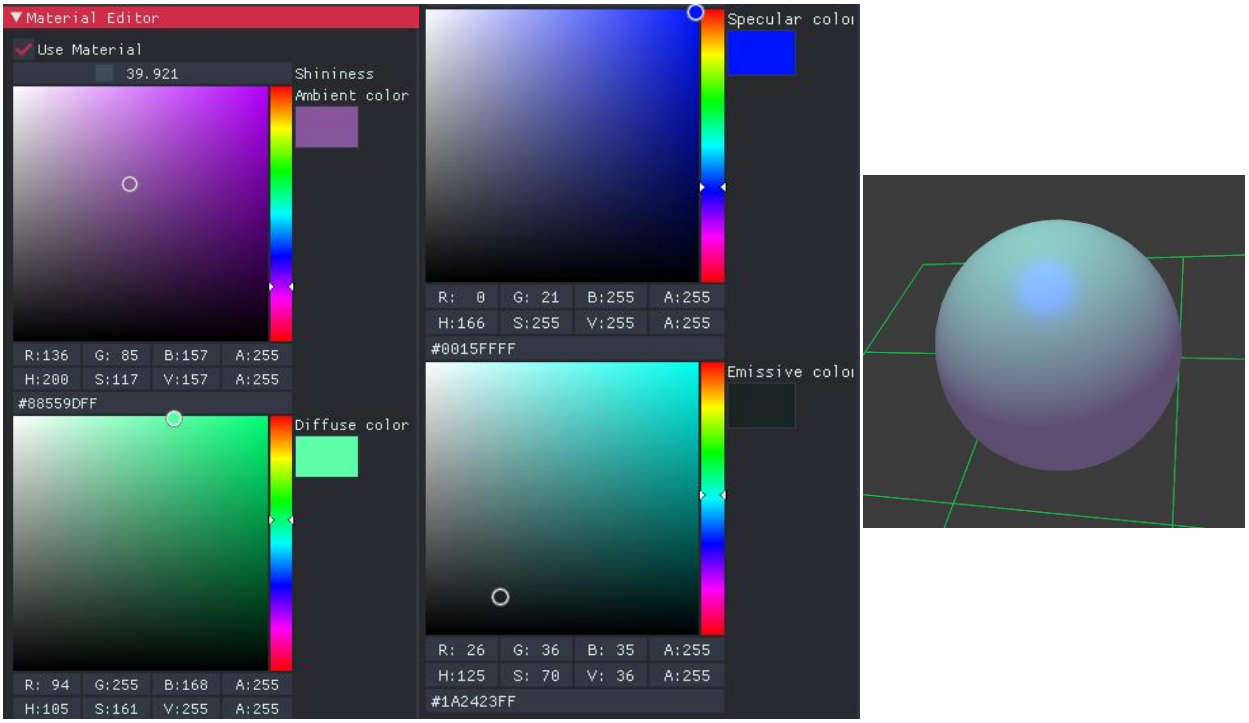
Le logiciel offre la possibilité de faire de l'échantillonnage d'image. Nous pouvons importer une image et se servir de celle-ci afin de générer des cubes dans la grille de la même couleur que la couleur du pixel au centre de chaque carré de la grille.

Si un utilisateur veut pouvoir dessiner son donjon sur une feuille avant ou bien se servir d'une vue de haut d'un donjon pour le recréer dans notre logiciel, il peut simplement importer l'image et ajuster la sensibilité dans l'interface graphique et ainsi décider s'il désire simplement dessiner les cubes noirs ou bien s'il veut inclure aussi toutes les autres couleurs possibles du système RGB.



1.4 Espace de couleur

Lorsqu'un élément est sélectionné, il est possible de modifier ses propriétés via le menu « Material Editor »



Le menu permet de personnaliser les éléments des objets tels que sa réflectivité, sa couleur ambiante, sa couleur diffuse, sa couleur spéculaire ainsi que sa couleur émise.

Les modifications sont propres à chaque objet et permettent donc de produire des thèmes dans les différentes pièces du donjon. Il est à noter que les modèles importés ne peuvent, actuellement, pas voir leur couleur modifiée. Le but serait de pouvoir leur ajouter des textures lors de la seconde partie du projet.

Dessin Vectoriel

2.1 Curseur dynamique

Pour les curseurs dynamiques, nous avons obtenu différents fichiers en ligne que nous avons par la suite modifiés à l'aide du logiciel Photoshop. Au niveau du code, nous avons créé une classe curseur permettant d'instancier les différents curseurs à l'aide d'une énumération et d'un « switch case ». Par la suite, nous avons adapté la forme du curseur (son état) selon les différentes actions réalisées dans la classe application :

mouseLeftPress pour la primitive vectorielle : curseur « cross »

mouseLeftPress et mouseRightPress : curseur « touch »

mouseLeftDragged et mouseRightDragged : curseur « drag »

Tous les mouseRelease : curseur « normal »

mouseMiddlePressed et mouseMiddleDragged : curseur « bucket »

2.2 Outils de dessin

Bien que plutôt basiques, nous avons implémenté quelques outils de dessin. Il est d'abord possible de modifier le « scale » de la grille ainsi que le nombre de colonnes et de rangées à l'aide du menu « Grid » de l'interface. De plus, comme mentionné dans la section « espace de couleur », il est possible de modifier la couleur de tous les éléments dessinés (à l'exclusion des modèles importés) à l'aide de l'outil de sélection de couleur.

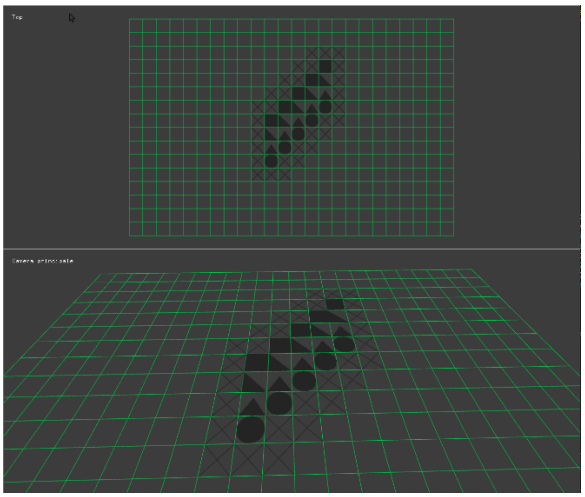
2.3 Primitives vectorielles *obsolète*

Nous avons ajouté une fonctionnalité qui permet à un utilisateur d'ajouter des primitives vectorielles servant de marqueur au sol. Il est possible d'ajouter les primitives vectorielles suivantes : la ligne, le rectangle, l'ellipse, un triangle rectangle et un triangle équilatéral.

Le but derrière la primitive vectorielle « ligne » est de dessiner un « X » au niveau du plancher pour permettre à l'utilisateur de se laisser une trace dans sa carte. Nous appelons la fonction « ofDrawLine » à deux reprises et donnons à chacune des lignes la position de deux vecteurs pour de sorte de dessiner un « X ».

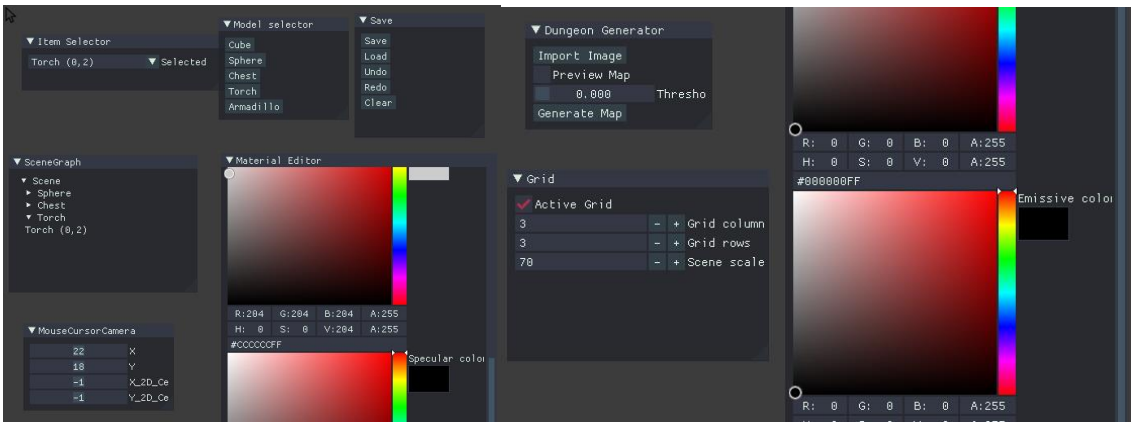
Le rectangle, l'ellipse, le triangle rectangle et le triangle équilatéral appellent les fonctions « ofDrawRectangle » « ofDrawEllipse » « ofDrawTriangle » auxquelles nous passons les positions de la souris en lien avec la grille pour dessiner les formes appropriées. Les applications

pour ces formes sont multiples ; de la délimitation de zone à la représentation d'objets et d'emplacements clés en passant par l'affichage d'indicateurs.



2.5 Interface

L'interface est produite à l'aide de la librairie ImGui. Bien que son intégration dans un projet « openFrameworks » se soit avérée difficile, le gain obtenu par son aisance d'utilisation et de sa personnalisation était plus que satisfaisant.



Son utilisation permet de créer des menus permettant un accès direct sur les différentes variables pour lesquelles nous désirons avoir une lecture en temps réel ou que nous désirons modifier.

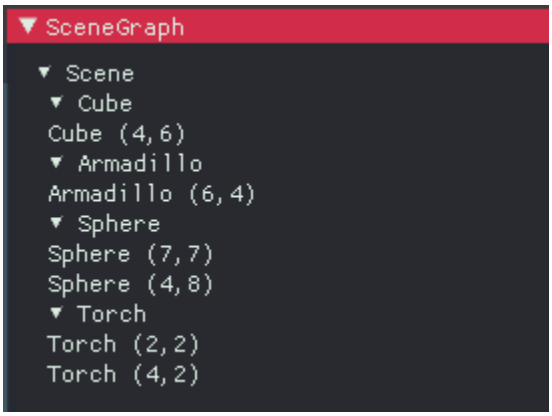
Le seul bémol est le manque de documentation sur certaines implémentations (voir la section « scene graph »). Cette problématique a été adoucie par l'existence de ChatGPT qui a beaucoup contribué à une documentation de fortune.

Transformation

3.1 Graphe de scène *obsolète*

La première version du graph de scène a été produite en utilisant un combo box à même le menu ImGui. Cela permettait d'avoir la liste des éléments et les rendait sélectionnables aisément.

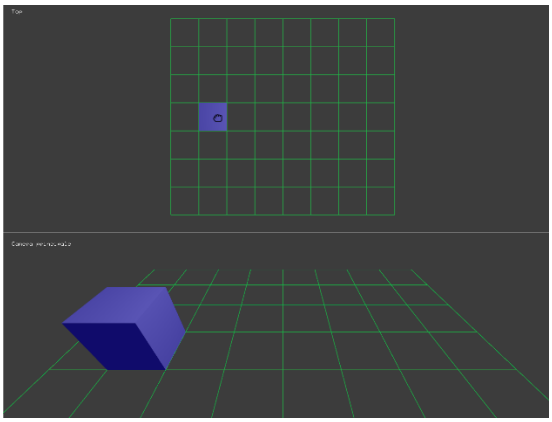
Cependant, le combo box ne permettait pas la représentation en arborescence des éléments. Cette problématique a été résolue en utilisant la fonction TreeNode de ImGui. Il ne suffit alors que de construire des nœuds parents qui sont nommés selon le type des objets. Les feuilles sont identifiées par le type suivi de leurs positions (x, z) dans la grille.

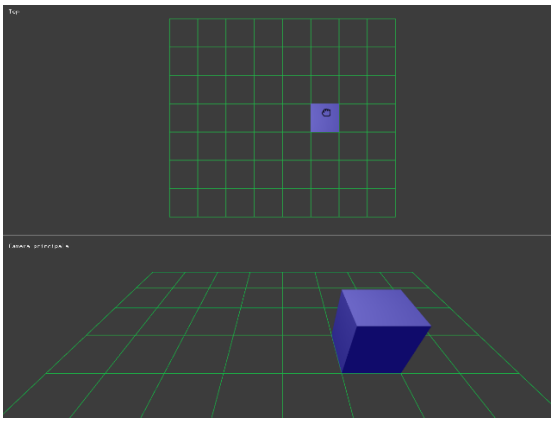


Nous avons choisi de conserver la première version comme un moyen de sélectionner un objet précis. Le modèle en arborescence nous permet d'avoir une vision d'ensemble de la scène.

3.3 Transformations interactives

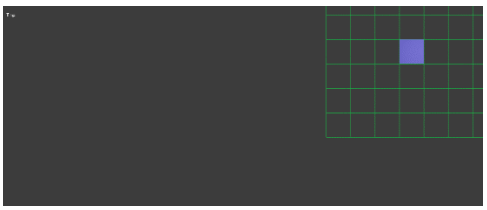
Afin de permettre une plus grande aisance dans la conception du donjon. Il est possible d'effectuer des translations des objets à l'aide du curseur de la souris.

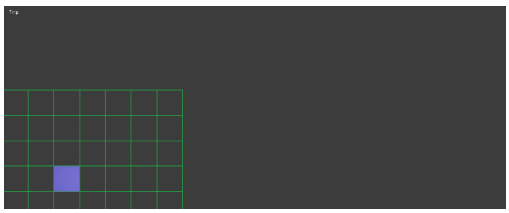




En maintenant le clic gauche du curseur enfoncé sur un élément de la fenêtre en vue orthogonale, il est possible de déplacer les objets et de visualiser le déplacement jusqu'au relâchement du clic.

Il est aussi possible de déplacer la grille à l'aide des flèches du clavier.





3.4 Historique de transformation

L'historique fonctionne en stockant les propriétés des objets sous forme de fichier Json contenant toutes les propriétés des transformations, ajouts et suppressions d'éléments. Ils sont ensuite stockés dans un vecteur d'historique. Les fonctions Undo et Redo restaurent ou suppriment les changements afin de modifier la scène selon l'historique enregistré.

Pour permettre la récupération des informations des Json des objets, ceux-ci contiennent :

- En premier lieu, une fonction toJson qui retourne un Json de tous les éléments pertinents à leur définition (position, couleurs, etc.)
- En second lieu, un constructeur qui prend en paramètre un fichier Json

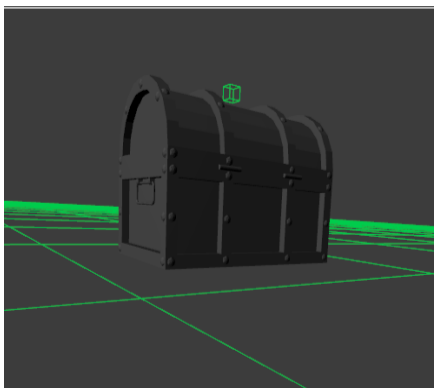
Selon les trois types de modifications (Création, Suppression, Modification), les opérations directes ou inverses sont effectuées afin de produire l'effet désiré. Il est à noter que les modifications de type Modifications ne sont que des opérations de Création et Suppression combinées.

Géométrie

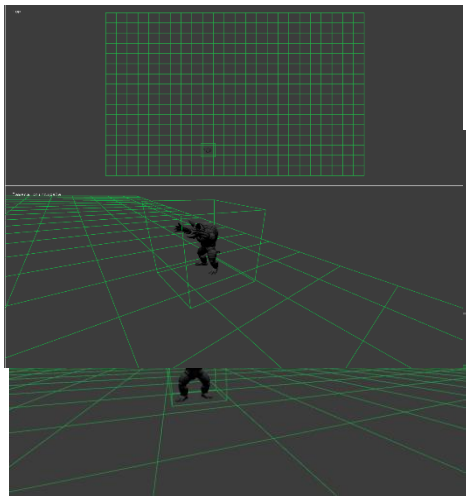
4.1 Boîte de délimitation

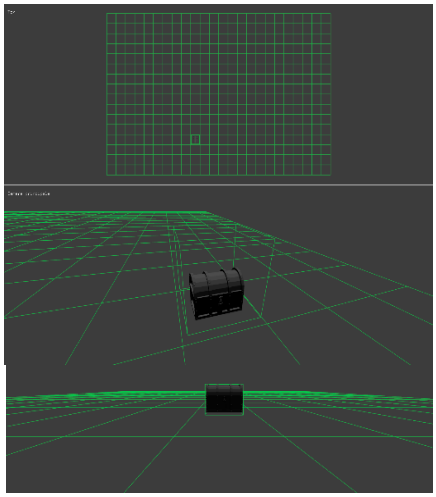
Pour la conception de la boîte de délimitation, nous avons pris la décision de procéder avec une fonction commune à tous les modèles 3D, permettant de dessiner un cube autour de ceux-ci. Cette partie nous a causé certaines difficultés, particulièrement en raison des origines des modèles.

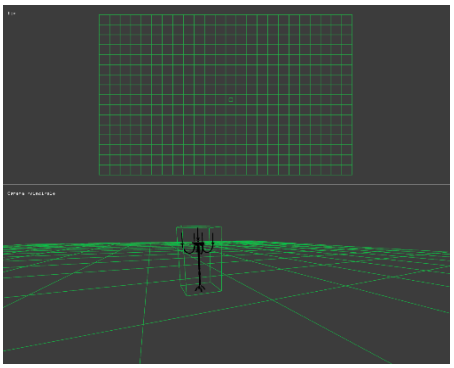
Nous avons utilisé une formule permettant d'obtenir la valeur du plus petit et du plus grand sommet du modèle et avons ensuite procédé au calcul de différence entre ces deux points. Initialement, nous avions oublié de multiplier les positions minimale et maximale par une matrice du modèle, ce qui nous donnait un intéressant résultat.



Une fois ce problème résolu, nous avons utilisé la différence mentionnée ci-haut comme paramètre servant à déterminer la taille de notre cube. Cependant, si la taille des boîtes dessinées s'ajustait au « scale » des modèles, elle ne se collait pas aux sommets de ceux-ci. Des ajustements manuels ont dû être faits dans le but d'ajuster les proportions de chacune des boîtes à leurs modèles.

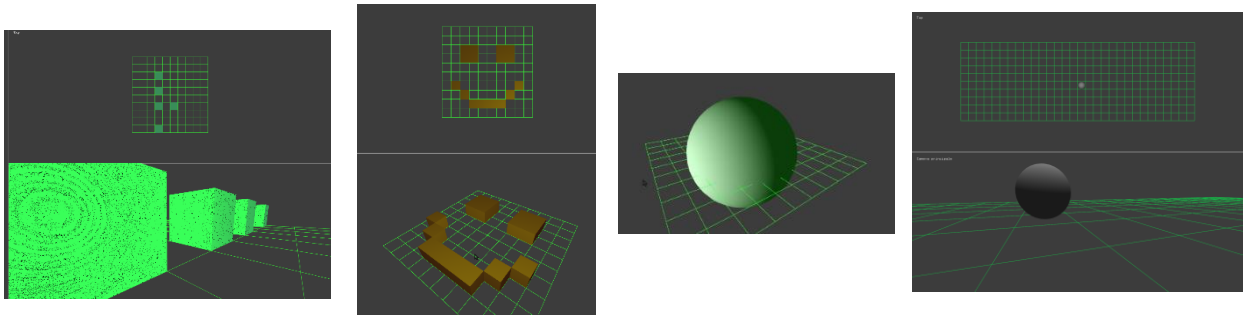






4.2 Primitives géométriques

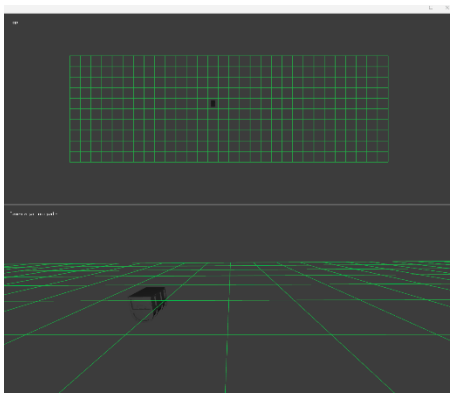
Pour cette partie, notre première primitive a été le cube, qui a servi à dessiner les murs du donjon. Nous avons fait en sorte que le cube se place directement sur les cases de la grille (se « snap ») dans le but de faciliter la façon dont il serait positionné sur la grille. Une fois nos tests initiaux réalisés avec un cube « hard codé », nous avons procédé à la création d'une classe Cube, implémentant une interface IObject, ce qui nous a permis d'aisément ajouter les cubes instanciés (ainsi que les autres objets créés par la suite) au graphe de scène. Ceci a été la base de l'interaction de tous les formes et modèles. Une fois le positionnement des cubes bien établi, nous avons procédé de la même façon avec une classe Sphère.

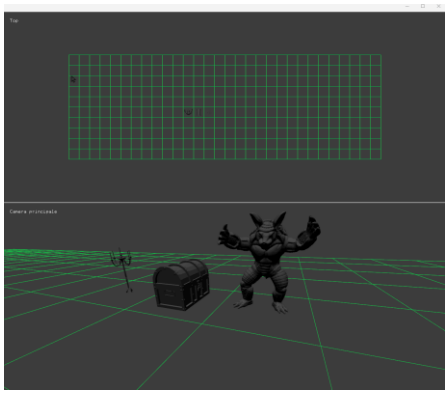


4.3 Importation de modèles

Nous avons pris la décision d'importer des modèles représentant des objets fréquemment retrouvés dans des donjons, soit une torche, un coffre et un ennemi. Pour ce faire, nous avons procédé au chargement initial des modèles dans le graphe de scène et avons créé des classes « Chest », « Torch » et « Armadillo » permettant de dessiner ces modèles. Dans le but de pouvoir être ajoutés au graphe de scène au même titre que les primitives 3D, nous avons basé lesdits modèles sur la même interface IObject servant au cube et à la sphère. Cela nous a permis de les gérer de la même façon que les cubes servant de murs, par exemple.

Il est à noter que nous avons rencontré une difficulté lors de l'importation, principalement au niveau de l'inversion du modèle. Le modèle se retrouvait en effet à l'envers au niveau de la grille, et une rotation en z provoquait un comportement inattendu (remise à l'endroit du modèle, mais visualisation de la position erronée à l'écran, malgré une bonne position enregistrée). Grâce au logiciel « Blender », nous avons pu identifier l'axe des y lors de la mise à l'échelle comme étant l'élément problématique. Un « setScale » négatif au niveau du « y » a permis de régler le problème.





Caméra

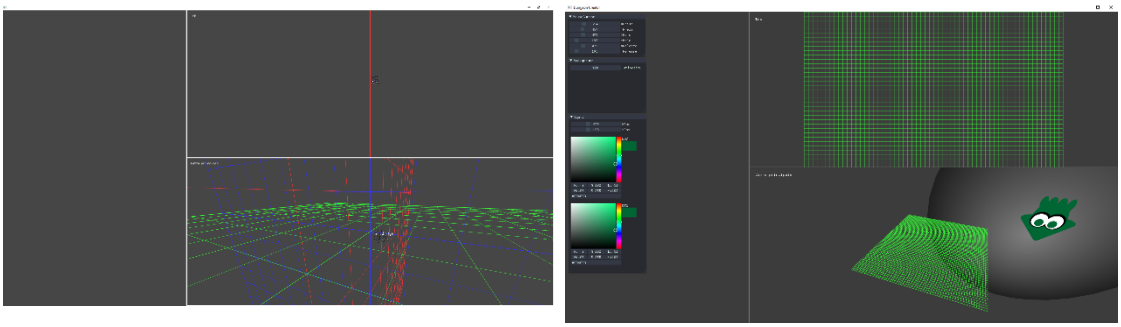
5.1 Point de vue

Deux caméras différentes sont présentes dans le logiciel. Celle en vue orthogonale, positionnée en haut à droite, est une OfCamera et celle en vue de perspective, positionnée en bas à droite, est une OfEasyCam. La OfEasyCam permet, de base, de déplacer la vue de la scène avec la souris en avec le « drag » des boutons et elle permet aussi de s'approcher et de s'éloigner de l'endroit où elle pointe avec la roulette de la souris. Pour ce qui est de la OfCamera, nous avons dû déclarer une fonction dans la classe « Cameras.h » qui déplace la « positionGlobal » de la caméra de cinq unités à chaque mise à jour de la classe « Application ». Pour s'assurer que cela n'affecte pas le positionnement des objets dans la grille de la vue de haut, il fallait aussi appliquer le même décalage de la caméra à la position où la souris pointe pour qu'elle puisse placer les objets au bon endroit.

5.2 Mode de projection

Il existe deux modes de projection dans notre logiciel; une projection en perspective qui est située en bas à droite et une projection orthogonale en haut à droite. Afin d'implémenter les caméras à deux endroits différents dans notre logiciel, nous avons dû implémenter ce qu'on appelle des « viewports » ; il s'agit simplement de OfRectangle que nous dessinons dans l'application. Au moment de faire les « begin() » des caméras, il faut alors passer en argument le « viewport » dans lequel nous voulons placer la caméra à l'intérieur.

Nous nous sommes basés sur des exemples présents dans les fichiers exemples d'« openFrameworks » afin de voir la façon d'implémenter les « viewports » et de leur associer une caméra.



5.3 Agencement

L'agencement se fait avec les deux caméras, car les deux caméras de notre logiciel affichent le rendu de la même scène en même temps, et ce dans un « viewport » unique par caméra.

Fonctionnalités TP2

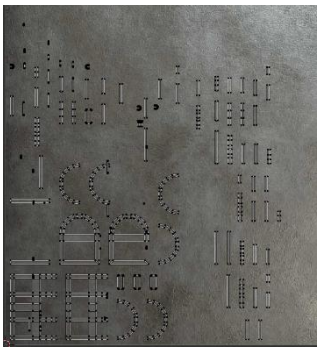
Texture

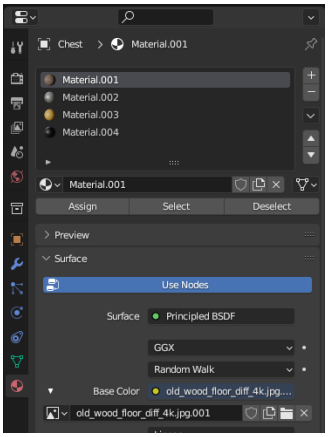
6.1 Coordonnées de texture

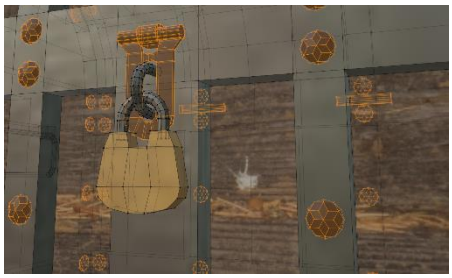
Un texturage manuel a été effectué sur la torche et sur le coffre, grâce au logiciel Blender. Initialement, une même texture avait été appliquée entièrement à la torche, cependant, cela ne pouvait être suffisant pour le coffre. De la recherche a donc été faite sur la façon d'appliquer différentes textures à différentes parties du modèle. Un travail très minutieux a été fait au niveau des différents sommets du modèle, pour « mapper » la texture adéquate à chacun des éléments.

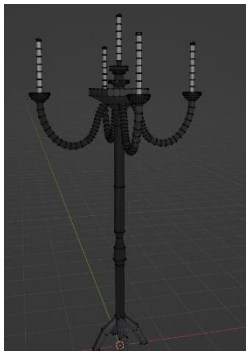
Par la suite, nous sommes revenus travailler au niveau de la torche, pour lui ajouter des chandelles séparées de la torche elle-même.

Finalement, les objets ont été exportés en fichier obj, et leur fichier .mtl a été modifié pour aller chercher les textures au bon endroit.









6.3 Mappage tonal

Grâce au shader fourni dans les exemples du cours, nous avons pu implémenter le mappage tonal dans notre projet de session. Il est imbriqué dans le shader de PBR que nous utilisons. Nous utilisons le mappage de Reinhard. Le mappage « aces filmic » a été retiré du Shader puisqu'il n'est pas utilisé dans notre projet.

6.5 Texture Procédurale

La texture servant à la base de la grille de jeu est générée par texture procédurale. Une texture est produite correspondant à la dimension de la grille. La couleur de chaque pixel est déterminée de façon aléatoire.

Afin de simuler un aspect de terre, les couleurs sont limitées entre les plages suivantes :

Rouge : 30% – 40%

Vert : 20% – 30%

Bleu : 10% – 30%

La composante aléatoire permet d'avoir une texture sans répétition précise.

Illumination classique

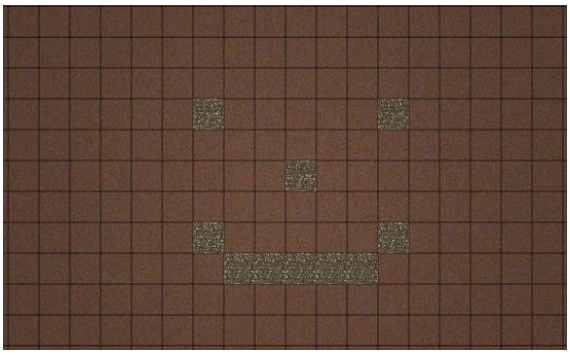
7.2 Matériaux

Les matériaux sont déjà implémentés dans notre logiciel depuis la remise du premier travail pratique. Ils sont attribués aux objet au moment que leur commande draw() est appelée et le tout peut être modifié avec un menu intégré dans notre logiciel.

7.3 Types de lumières

Nous avons instauré les quatre types de lumières suivant :

Ambiante : Utilisée pour l'environnement de base de la scène



Directionnelle :

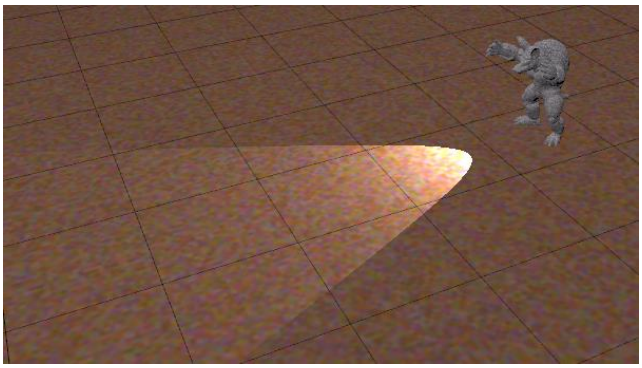
Un éclairage directionnel provenant du nord est produit, permettant aux Armadillos de ne jamais avoir le soleil dans les yeux.

Exemple de face :

Exemple de dos :

Ponctuelle : Utilisée pour les torches et sphères

Projecteur : Utilisé pour le champ de vision de Mr. Armadillo

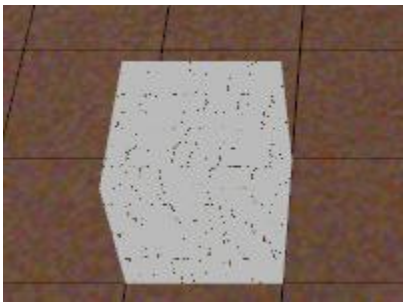


7.4 Lumières multiples

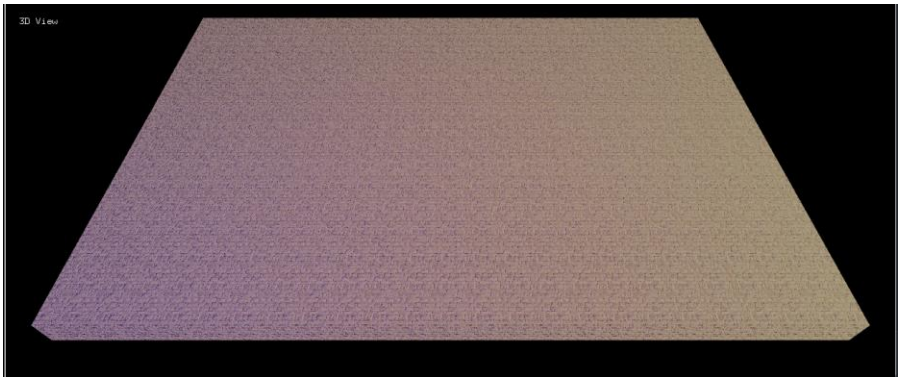
Initialement, un shader de lumières multiples traitant tous les types de lumière a été réalisé (il se trouve toujours au niveau de la section shader du projet, même s'il est inutilisé). Cependant, suite à la nécessité d'utiliser le shader PBR pour traiter la métallicité et le « roughness » des objets, une modification servant à traiter plusieurs lumières a été ajoutée dans celui-ci. De plus, l'attribution des valeurs d'atténuation a été déplacée dans l'application dans le but de les rendre modifiables.

Par la suite, nous avons fait en sorte que les informations des quatre premières torches placées dans la scène soient envoyées au shader pour être passées comme lumières dynamiques. Il est cependant à noter que nous avons rencontrer un problème au niveau de la caméra qui rendait faisait en sorte que l'objet le plus près de la caméra était toujours plus éclairé.

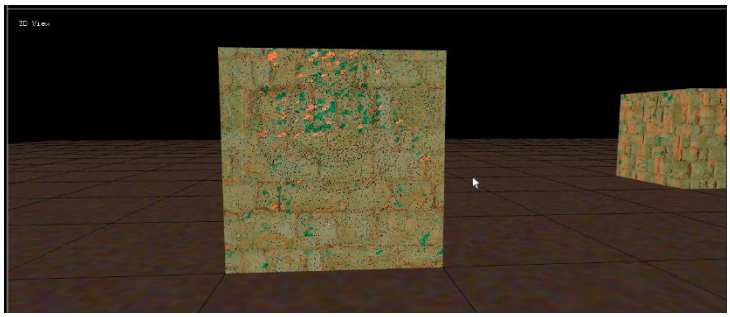
Problème d'atténuation initial :



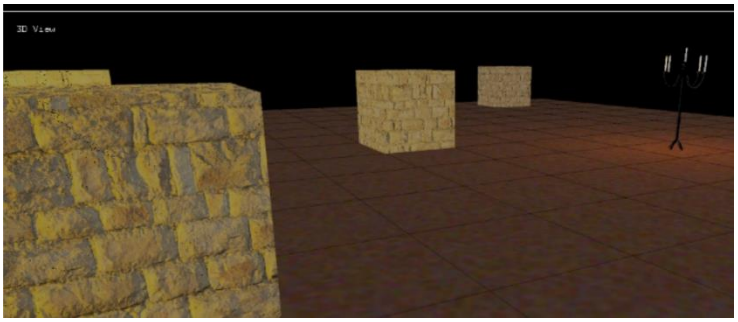
Première passe de lumières multiples (bleu en bas à gauche, jaune à droite, rouge au centre, blanche sur la souris)



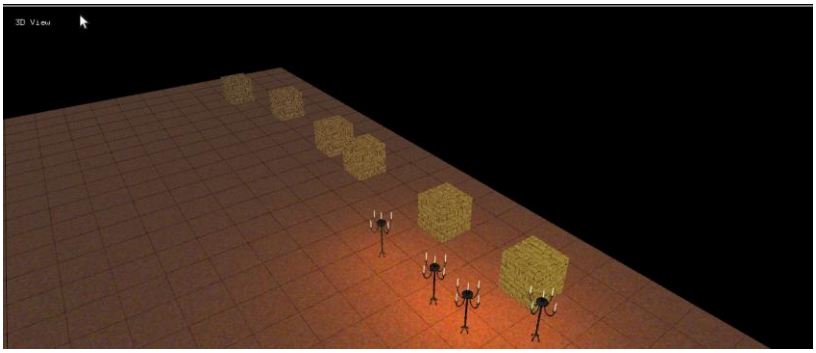
Première tentative avec les torches (étrange pixélisation en raison d'un problème de GPU)



Premier bon résultat :



Résultat final :



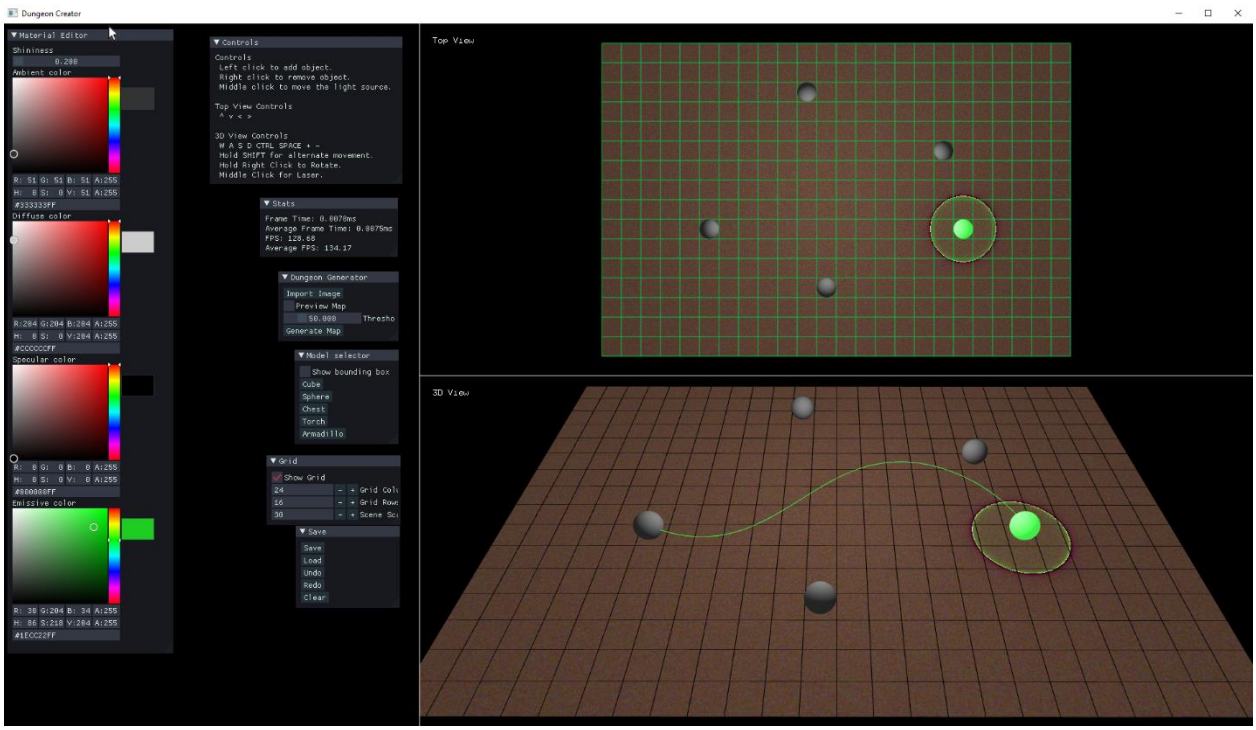
Topologie

8.1 Courbe paramétrique

Pour l'ajout d'une courbe paramétrique, nous nous sommes servis de la formule de Bézier. Une fois que cinq sphères sont placées, la courbe est calculée, ce qui permet de la faire apparaître. Nous nous sommes assurés que la formule prenait en compte cinq positions différentes afin de respecter la demande pour le livrable. Si plus de cinq sphères sont ajoutées, les cinq premières servent à tracer la courbe. Si l'une des premières sphères est supprimée, la formule prend la sphère suivante (la sixième) en compte.

Voici la formule utilisée pour calculer une courbe de Bézier avec cinq points :

glm::vec3 point = pow(1 - t, 4) * p0 + 4 * pow(1 - t, 3) * t * p1 + 6 * pow(1 - t, 2) * pow(t, 2) * p2 + 4 * (1 - t) * pow(t, 3) * p3 + pow(t, 4) * p4;

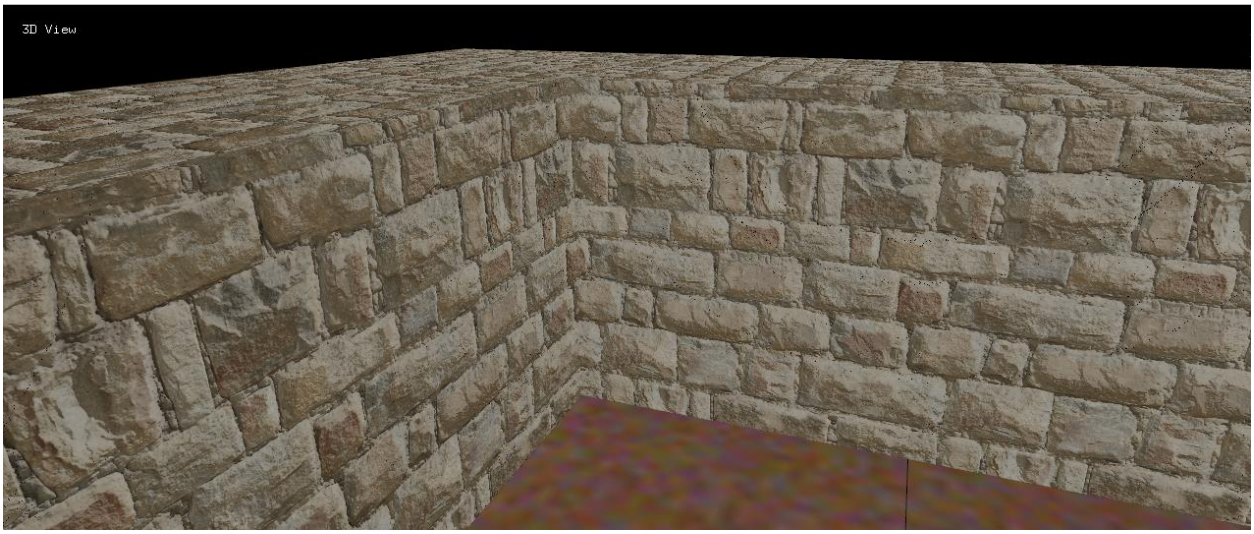


Et comme on le remarque dans l'image placée ci-haut, la courbe prend la même couleur que la dernière sphère qui a été placée pour former la courbe. Cependant, depuis l'ajout du shader, nous avons dû donner une couleur fixe à la courbe, car elle n'était plus influencée par la couleur du matériel de la sphère. Elle est maintenant blanche.

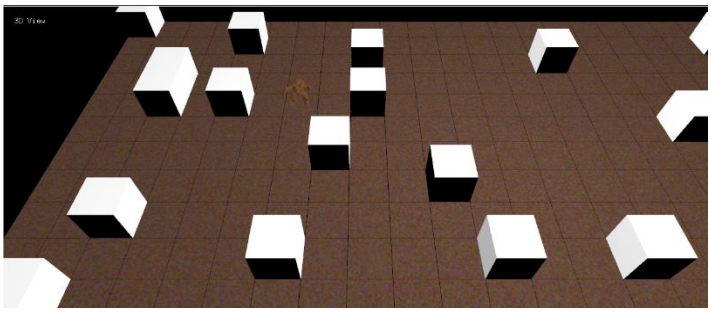
8.5 Effet de relief

Un « normal mapping » a été effectué au niveau de la texture du cube. Initialement, un shader servant exclusivement au « normal map » a été développé. Plusieurs versions du shader avec plusieurs algorithmes ont initialement été testées dans une tentative de faire fonctionner le tout ; une erreur au niveau de « texcoord » s'était glissée, faisant en sorte qu'aucune des versions ne fonctionnait. Une fois l'erreur corrigée, chacun des shaders (Lambert, Phong, une version inspirée de « Learn OpenGL », une version inspirée de « leozimmerman » sur GitHub, une version inspirée d'une discussion de StackOverflow et même une version concoctée par ChatGPT (après un bon moment à essayer des choses, il faut avouer que nos tentatives se sont trouvées à être de plus en plus farfelues).

Finalement, la version finale a été celle implémentée directement dans le shader PBR fourni en exemple du cours, en ajoutant une normal map en provenance de Poly Haven.



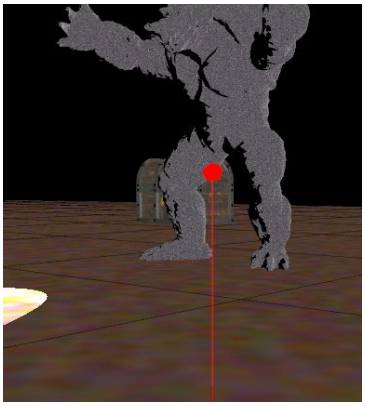
Problème lors de l'implémentation du shader :

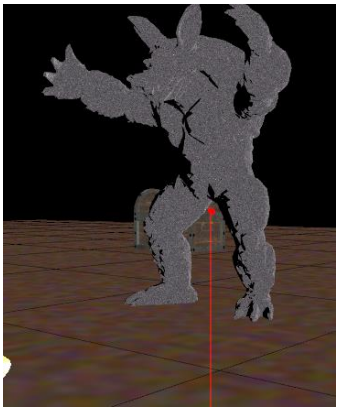


Lancer de rayon

9.1 Intersection

Nous avons fait une classe RAY inspiré de ofxRaycaster avec laquelle nous pouvons configurer un rayon en lui passant le curseur de la souris, une caméra, un viewport et une liste d'objets. Ensuite un rayon est défini comme partant de la caméra et allant vers l'infini en direction de la souris. La liste d'objet est ensuite traversée pour savoir s'il y a une intersection avec le « bounding box » de chaque objet. Si une intersection est trouvée, l'index de l'objet est retourné. Il y a une deuxième méthode qui sert à faire le dessin d'un rayon dans la scène. Celle-ci utilise la même technique, mais les objets dont les « bounding box » sont sur la trajectoire du rayon sont ensuite passés dans un algorithme de détection de collision avec chacun des « meshes » afin de trouver exactement où devrait s'arrêter le laser. Une sphère est alors dessinée à cet endroit. L'idée d'isoler d'abord les objets potentiellement en collision via le « bounding box » est assez importante en termes de performance, car valider les collisions avec tous les « meshes » d'une scène peut potentiellement être très lourd.





Illumination moderne

10.1 HDR

Grâce au shader fourni dans les exemples du cours, nous avons pu implémenter le calcul dans l'espace HDR dans notre projet de session. Il est imbriqué dans le shader de PBR que nous utilisons.

10.3 Métallicité

Grâce au shader fourni dans les exemples du cours, nous avons pu implémenter le calcul en lien avec la métallicité de la matière dans notre projet de session. Il est imbriqué dans le shader de PBR que nous utilisons. Puisque le shader est appliqué à nos murs – qui ont une texture de pierre – un facteur de métallicité de 0.3 a été appliqué, dans le but d'offrir le rendu le plus réaliste possible.

Métallicité de 1

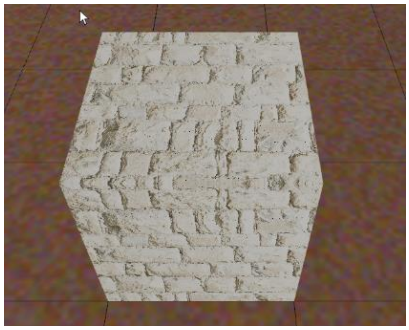
Métallicité de 0 :

10.4 Microfacettes

Grâce au shader fourni dans les exemples du cours, nous avons pu implémenter le calcul en lien avec les microfacettes dans notre projet de session. Il est imbriqué dans le shader de PBR que nous utilisons. Tout comme la métallicité, puisque le shader est appliqué à nos murs, un facteur de rugosité de 0.9 a été appliqué. Des tests visuels ont été effectués dans le but de trouver le facteur le plus réaliste pour notre projet.

Rugosité de 1 :

Rugosité de 0 :



Bonus

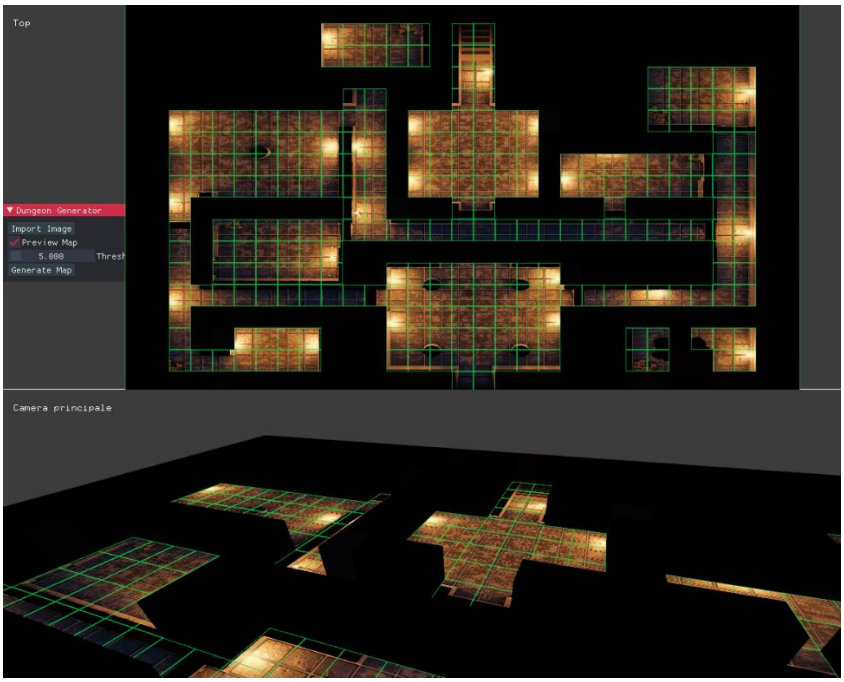
Livrable 1

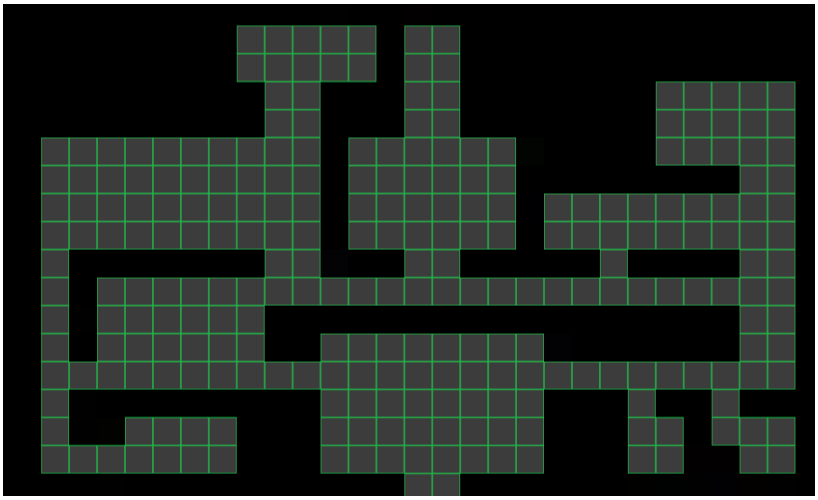
Carte auto constructrice à partir d'une image

Nous avons inclus un outil qui, à la suite de l'importation d'une image :

- Pixelise l'image selon les dimensions de la grille de jeu.
- Produit les murs du donjon selon une densité (dont le seuil est ajustable)

Un mode « Preview » est disponible avant de faire la modification définitive de la grille de jeux.





Persistance

Il y a la possibilité de sauvegarder le donjon à l'aide d'un bouton SAVE. L'enregistrement produit des fichiers .json dans le répertoire …/bin/data/Save.

Le fichier appdata.json contient les propriétés de la grille de jeu alors que objects.json contient les propriétés des objets positionnés sur la grille.

Le bouton LOAD récupère les deux fichiers .json et construit la grille ainsi que ses composantes selon leurs instructions.

Source lumineuse mobile

Nous avons fait l'ajout d'une source de lumière mobile qui suit le mouvement de la camera3D.

Personnalisation des objets 3D

En plus de la couleur, nous avons à ajouter des éléments de paramétrisation supplémentaire :

- La possibilité d'ajuster le niveau de réflectivité des objets
- Sélectionner des couleurs multiples sur un même objet :
    - Ambiante
    - Diffuse
    - Spéculaire
    - Émise

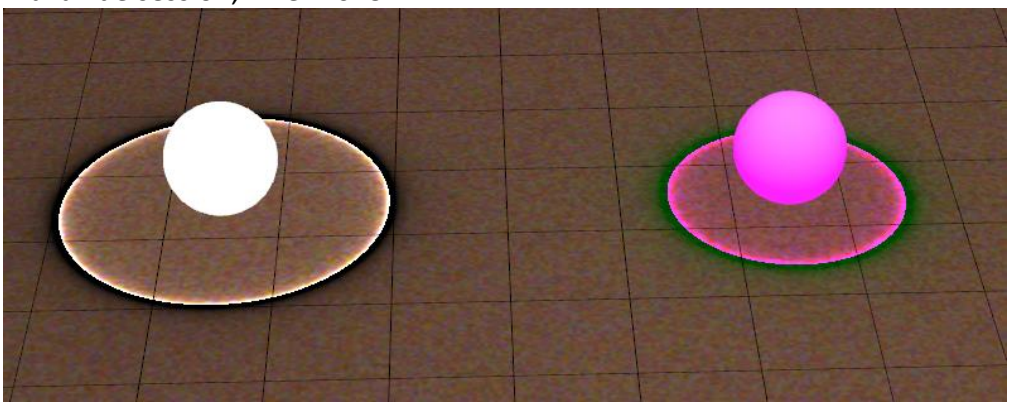
Livrable 2

Effet lumière avec aura de couleur inverse

L'effet lumineux des sphères est produit en utilisant une atténuation variable selon le nombre de "frame".

Un effet de progression et régression est produit en suivant le nombre de cycles depuis la création de la sphère. Nous avons découvert qu'en utilisant une valeur négative sur l'atténuation quadratique de la lumière, cela produisait une inversion de couleur aux limites de la lumière.

Cette approche n'est pas un usage normal prescrit pour l'atténuation, mais cela permet un effet unique pouvant produire une lumière noire.



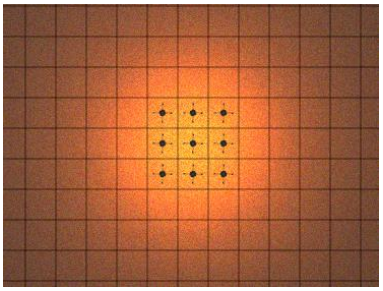
Effet de scintillement de flame

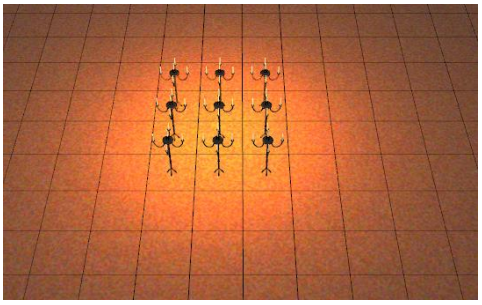
Afin d'apporter du réalisme à la lueur produite par les torches, nous avons créé un algorithme aléatoire de couleur produite par les flammes.

La teinte est produite en ayant 43% de vert pour 100% de rouge.

Une valeur aléatoire d'intensité de rouge est déterminée entre 73% et 100

A chaque "frame", le niveau de rouge est déterminé et cela produit un effet de scintillement de flamme (il est ici possible de voir le « halo » créé par l'effet de scintillement)





Ressources

TP1 :

Curseurs

Les curseurs ont été obtenus à partir de site internet. Malheureusement, à la suite de l'abandon de l'un de nos membres, nous ne sommes pas en mesure de retrouver la source où quatre d'entre eux ont été obtenus. Cependant, le cinquième a été obtenu du site suivant :

Croix : https://thenounproject.com/icon/cursor-cross-pointer-1881152/

Primitives 2D

Les primitives 2D ont été créées à l'aide des fonctions d'« openFrameworks »suivantes :

- Carré : ofDrawRectangle
- Cercle : ofDrawEllipse
- X : ofDrawLine
- Triangles : ofDrawTriangle

Primitives 3D

Pour les primitives 3D, nous avons travaillé avec les fonctions de « openFrameworks » suivantes :

- Cube : ofdrawBox
- Sphère : ofdrawSphere.

Modèles 3D

Pour les modèles 3D nous avons utilisé les sources suivantes

Armadillo: https://123free3dmodels.com/3d-armadillo-monster-creature-model-151695

Coffre : https://www.cgtrader.com/free-3d-models/interior/house-interior/treasure-chest-low-poly-pbr-86a76875-a1cb-425e-98f2-66b65852c3ea

Torche : https://www.cgtrader.com/free-3d-models/interior/other/dungeon-castle-assets-pack (modèle LargeCandle)

TP2 :

Illumination

Le shader d'illumination multiple a été créé suite à la réalisation du tutoriel sur l'illumination du site « Learn OpenGL ». Le tutoriel « Lighting » entier a été réalisé, et le résultat final s'apparente au shader trouver dans la section « Multiple Light », tout en étant adapté à la structure avec openFrameworks.

https://learnopengl.com/

Textures :

Murs : https://polyhaven.com/a/rustic_stone_wall

Coffre bois : https://polyhaven.com/a/old_wood_floor

Coffre métal pâle : https://free-3dtextureshd.com/download/metal-aluminium-texture-seamless-bpr-material-high-resolution-free-download-hd-4k/

Coffre métal foncé : https://images-wixmp-ed30a86b8c4ca887773594c2.wixmp.com/f/ab371d58-f694-4953-a2e5-c79acedd9f56/dbqqe9i-47d899b5-caac-4e03-ac53-a7e8506c39e2.jpg?token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1cm46YXBwOjdlMGQxODg5ODIyNjQzNzNhNWYwZDQxNWVhMGQyNmUwIiwiaXNzIjoidXJuOmFwcDo3ZTBkMTg4OTgyMjY0MzczYTVmMGQ0MTVlYTBkMjZlMCIsIm9iaiI6W1t7InBhdGgiOiJcL2ZcL2FiMzcxZDU4LWY2OTQtNDk1My1hMmU1LWM3OWFjZWRkOWY1NlwvZGJxcWU5aS00N2Q4OTliNS1jYWFjLTRlMDMtYWM1My1hN2U4NTA2YzM5ZTIuanBnIn1dXSwiYXVkIjpbInVybjpzZXJ2aWNlOmZpbGUuZG93bmxvYWQiXX0.M3ACw_14A4a8PLM3GzFjU8S_2T-5u_rpepMi0CbNP1Q

Coffre laiton : https://nielsharbo.dk/resources/tileabletextures/steambrass.jpg

Torche métal : https://media.istockphoto.com/id/460230555/photo/a-black-metal-texture-background.jpg?s=612x612&w=0&k=20&c=NhUvxOxp4PDijHNFUDzyAy3LlkprtuYvPltibP9ev_Q=

Torche chandelles : https://media.istockphoto.com/id/481266709/photo/wax-texture.jpg?s=612x612&w=0&k=20&c=qAfBMvF1KNwpDmGjvLpXzoFRvrVMKTBe2PiK2TkP9s0=

Armadillo : https://polyhaven.com/a/aerial_asphalt_01

Shaders

Lors des tests pour la réalisation du normal mapping, plusieurs ressources ont été consultées dans le but de réussir l'implémentation d'un shader de normal mapping.

Test1 : https://learnopengl.com/

Test2 :https://github.com/leozimmerman/ShadersLibrary/blob/master/2-NormalMapping/bin/data/normMap.frag

Test3 : https://forum.openframeworks.cc/t/tangent-of-a-normal-in-an-ofprimitive/25100/11

Test4 : http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/

Test5 : https://chat.openai.com/

De plus, le shader de PBR implémenté dans le programme est une version modifiée du Shader PBR fourni en exemple dans le module 10 du cours.

Présentation

L'équipe est composée de David Gaudet, Didier Michaud, Laurie Théberge et Patrick Veilleux. Chacun des membres s'est occupé d'une partie du travail en particulier, même si plusieurs points ont été révisés en sous-équipes.

David Gaudet :

TP1 :

- Rédaction du document design
- Implémentation du graphe de scène
- Implémentation de scene data
- Implémentation des fonctions de persistance (Save, Load, Undo, Redo)
- Implémentation de la grille de jeux
- Implémentation de l'algorithme de positionnement du curseur dans la fenêtre 2D

TP2 :

- Création de la texture procédurale
- Création de l'illumination :
    - Directionnelle
    - Torches (lumière scintillante)
    - Sphères (lumière pulsante)
    - Armadillo (projecteur pour « champ de vision)
- Assistance pour la courbe
- Assistance pour l'illumination
- Rédaction du document design

Didier Michaud :

TP1 :

- Implémentation de OfxImgui pour le GUI
- Transformation de la position des objets avec l'event mouseDragLeft
- Déplacement de la caméra du haut avec les flèches du clavier
- Implémentation et gestion de la caméra
- Création de la boîte de délimitation
- Implémentation des primitives vectorielles

TP2 :

- Ajout d'une courbe paramétrique
- Aide sur le Shader d'illumination
- Aide sur le Shader de normal map
- Instances des lumières dynamiques dans l'application
- Rédaction du document design

Laurie Théberge :

TP1 :

- Implémentation des modèles 3D
- Sélection des formes à partir du GUI
- Implémentation de la boîte de délimitation
- Options d'affichage des boîtes de délimitation
- Création des primitives vectorielles (première version)
- Création et rédaction du document design

TP2 :

- Création des shaders
    - lumières multiples
    - normal map
- Modification du shader de PBR avec lumières multiples et normal map
- Compléter l'implémentation du shader dans l'application
- Réalisation du texture mapping manuel du coffre et de la torche
- Rédaction du document design

Patrick Veilleux :

TP1 :

- Architecture du code
- Rédaction du document design
- Conception de l'algorithme de pixélisation des images
- Complétion du undo-redo (compatibilité avec couleurs)
- Réalisation du vidéo

TP2 :

- Algorithme de lancer de rayons (intersection)
- Amélioration de Iobject pour réduire la répétition de code
- Amélioration de la caméra pour contrôles se rapprochant semblable aux jeux vidéo
- Réalisation du vidéo