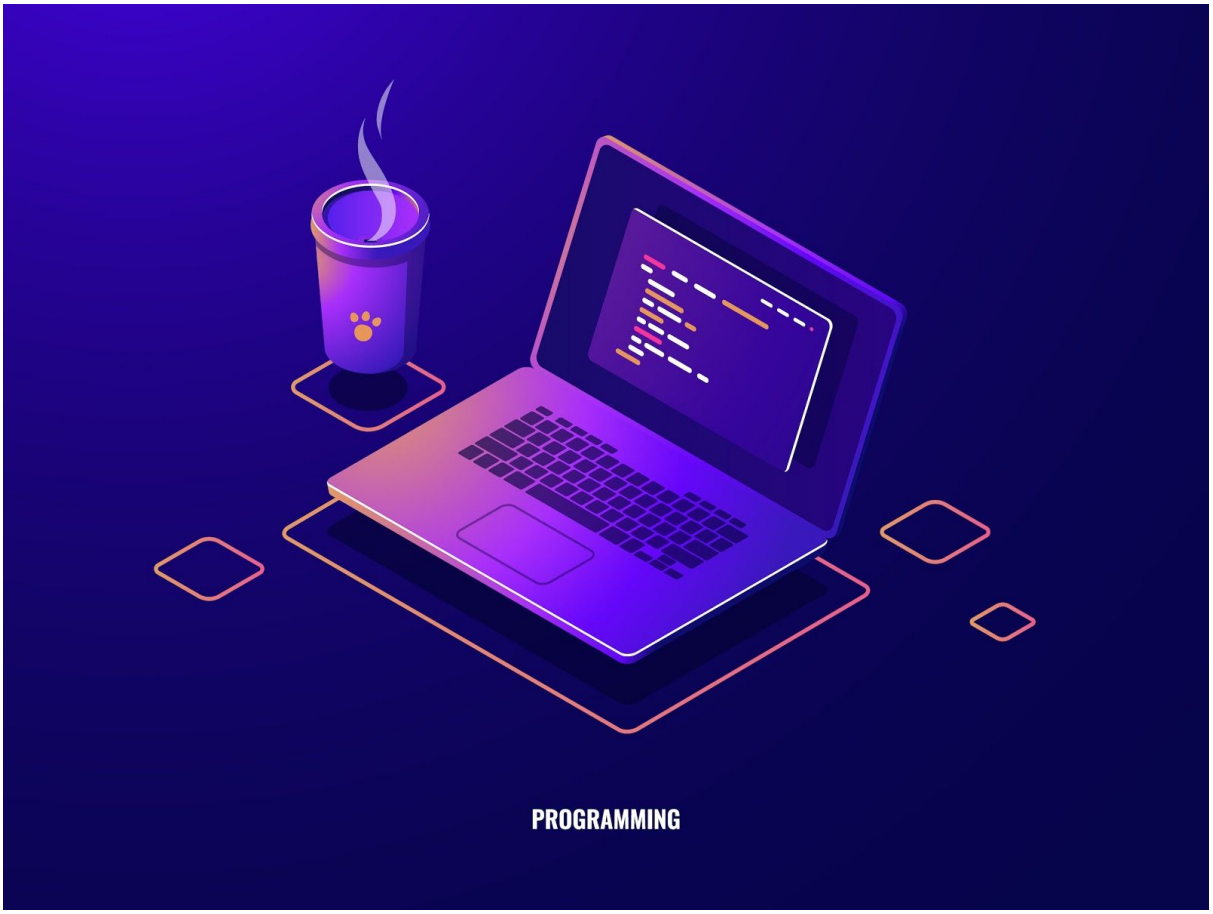
Cyrielle LASSARRE
Océane RIOSSET

WORLD IMAKER

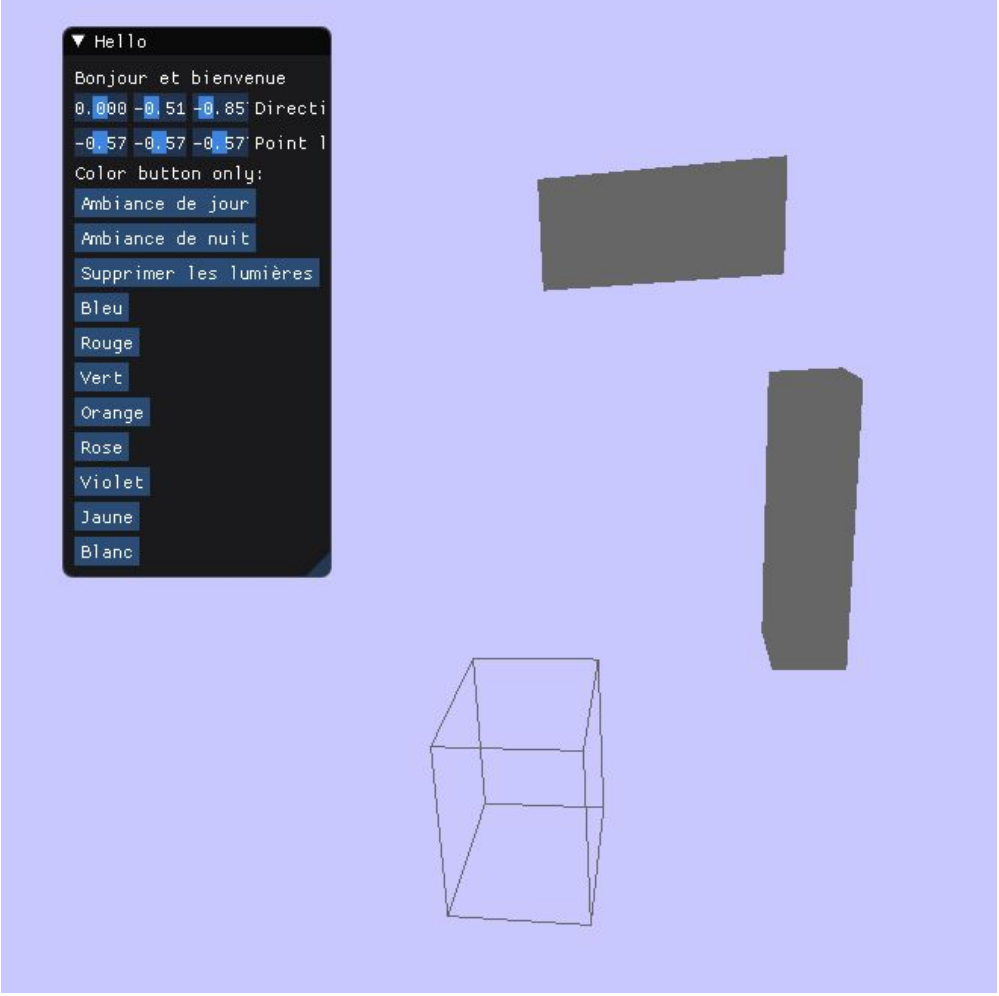


Étudiantes en IMAC 2
Année universitaire 2019-2020

INTRODUCTION

Dans le cadre des cours de mathématiques et de synthèse d'images, nous avons réalisé un Word IMaker qui est un éditeur-visualiseur de terrain et de scène en 3D. Un certain nombre de fonctionnalités ont été implémentées selon les consignes données afin de permettre à l'utilisateur de créer un monde à partir de blocs cubiques et de naviguer à l'intérieur de ce dernier.

Nous avons aidé du mieux que nous pouvions notre ami Toto qui nous a commandé ce Word IMaker. Nous espérons qu'il sera conquis par nos travaux.



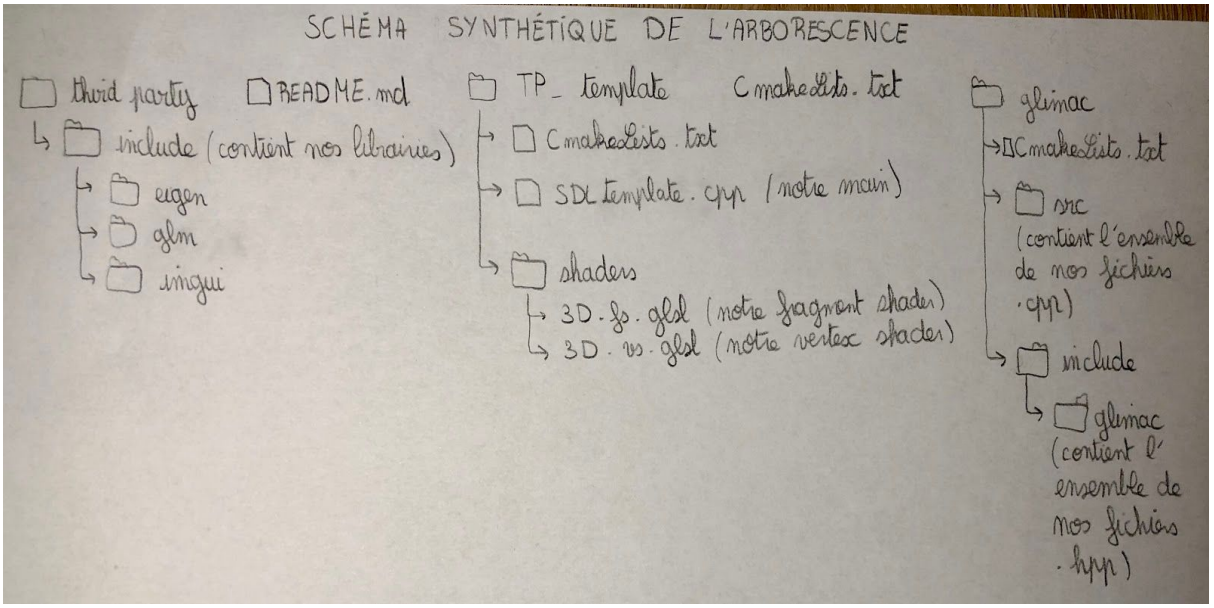
I. FONCTIONNALITÉS

FONCTIONNALITÉS	ENTIÈREMENT	PARTIELLEMENT	PAS FAIT
Affichage d'une scène avec des cubes	**X**		
⇒ *État initial*	*X*		
⇒ *Shaders*	*X*		
Édition des cubes	**X**		
⇒ *Curseur*	*X*		
⇒ *Contours*	*X*		
⇒ *Couleurs*		*X* changer la couleur d'un cube à la fois	
⇒ *Menu*	*X*		
Sculpture du terrain	**X**		
⇒ *Ajouter un bloc*	*X*		
⇒ *Supprimer le bloc*	*X*		
⇒ *Extruder ou creuser le terrain*	X		
Génération procédurale	**X**		
Ajout de lumières	**X**		
⇒ *Ajouter une lumière ponctuelle*	*X*		
⇒ *Ajouter une lumière ambiante*	*X*		
⇒ *Ajouter une lumière ambiante et une lumière ponctuelle de manière simultanée*	*X*		
⇒ *Supprimer une lumière*	*X*		

Fonctionnalités additionnelles			**X**
⇒ *Outils de painting*			*X*
⇒ *Texture des blocs*			*X*

II. ARBORESCENCE DE NOTRE PROJET

Afin de faciliter la compréhension de ce rapport et de notre projet, nous avons établi ce schéma synthétique. Il montre les différents dossiers et fichiers de l'organisation de notre World Imaker.

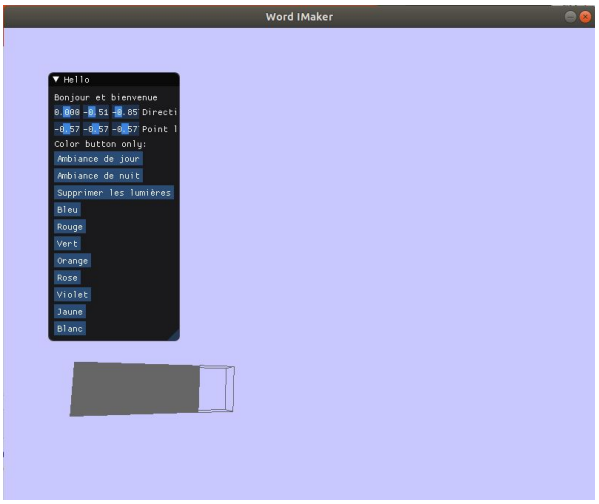


III. IMPLÉMENTATIONS

a) Affichage d'une scène avec des cubes

Pour l'affichage d'une scène, nous devions d'abord pouvoir créer un cube.
Pour cela, nous avons implémenté une classe cube qui contient tous les attributs et méthodes nécessaires (inclus l'initialisation des buffers). Pour dessiner le cube, nous avons entré les coordonnées des sommets du cube auxquels on associe des indexs. On dessine les faces à partir des points rentrés dans l'ordre des indexs.
Pour la caméra, nous avons choisi d'implémenter une caméra Freefly qui peut donc se déplacer dans tous les sens de rotation. Pour cela, nous avons fait changer la position de la caméra grâce à des coordonnées sphériques que l'on fait varier.

Nous avons aussi fait un shader dans lequel nous initialisons les vertexs pour la position et la couleur.



b) Édition des cubes

Sélectionner les cubes

"J'ai toujours rêvé de voyager dans l'espace" rêvasse Toto.

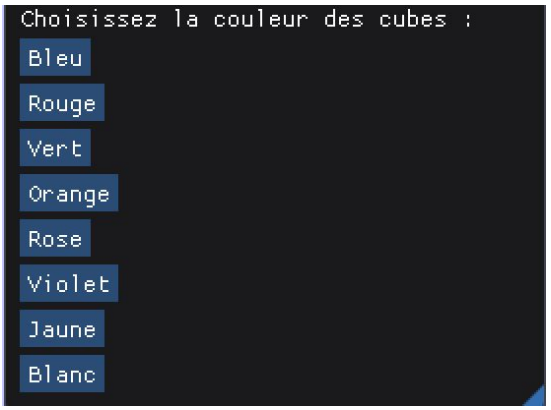
L'édition du terrain nécessite un curseur pour pouvoir sélectionner les cubes. Pour se faire, nous avons repris un cube dont nous avons seulement tracer les contours. Il peut se déplacer sur l'axe x (appuyer sur la touche x et les flèches gauche/droite du clavier), sur l'axe y (pareil mais appuyer sur la touche y) et sur l'axe z (appuyer sur la touche z). Il peut donc bouger dans toutes les directions de l'espace 3D : haut, bas, gauche, droite, avant et arrière.

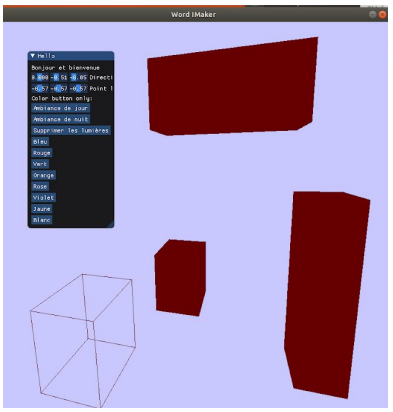
Les couleurs

"Vis ta vie en couleur c'est le secret du bonheur" chantonne discrètement Toto.

Pour implémenter les couleurs et ainsi pouvoir créer différents types de cubes, nous avons utilisé les shaders pour déclarer les variables nécessaires et la bibliothèque ImGui où nous avons pu attribuer différentes valeurs RGB comprises entre 0 et 1.

Une fois que nous avons implanté la possibilité de changer les couleur des cubes, nous avons décidé de donner la possibilité à l'utilisateur de choisir entre plusieurs couleurs depuis notre scène sans avoir à modifier les valeurs dans le code. Nous avons donc créé des boutons grâce à la librairie ImGui avec des couleurs prédéfinies.



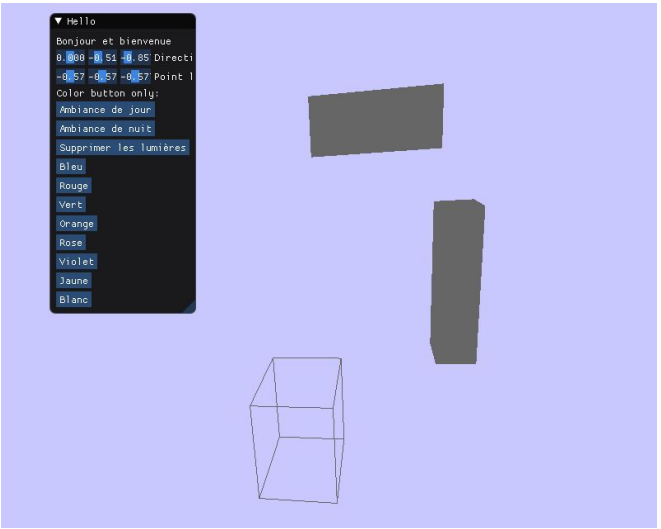


c) Sculpture du terrain

La topologie du terrain est modifiable : c'est maintenant que notre projet devient réellement un éditeur de cubes.

On peut en effet ajouter un cube dans la scène, en "envoyant" les coordonnées du curseur comme les coordonnées du cube à ajouter. Il suffit de se positionner avec le curseur au bon endroit sur l'axe des x, y et z puis d'appuyer sur la touche return.
Nous avons aussi la fonctionnalité de supprimer les cubes, il suffit de se positionner sur celui souhaité et d'appuyer sur la touche Backspace. En faisant cela, on récupère la position du cube, on place l'élément à retirer à la fin de la liste des cubes et on retire ce dernier élément.

Nous avons aussi implémenté les fonctionnalités dig et extrude. La position x du curseur est récupéré et on teste pour savoir si on a bien une colonne de cubes à cette position, si oui on se positionne tout en haut de la colonne. Ensuite, soit on appuie sur la touche E et cela "extrude" (donc on ajoute le cube en haut de la colonne), soit on appuie sur la touche D et cela "dig"(donc on supprime le cube en haut de la colonne).



d) Génération procédurale

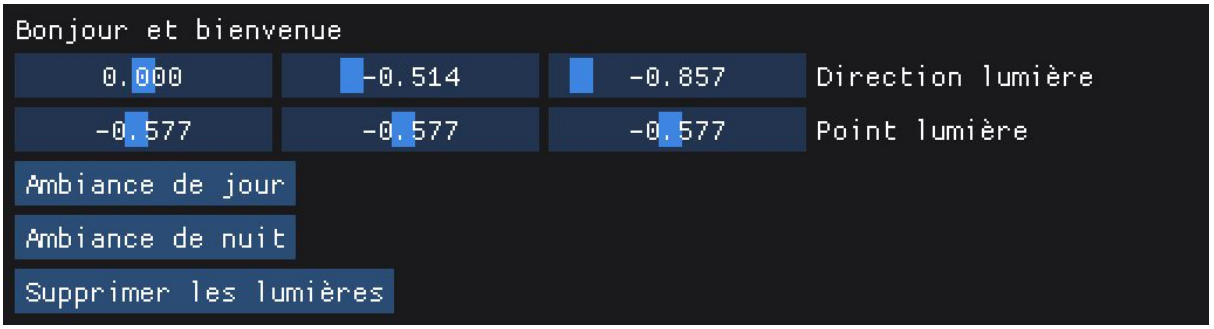
Pour la génération procédurale, nous avons fait une matrice avec les points de contrôle à laquelle nous appliquons la fonction radiale choisie. Pour cela, on cherche les omégas de la fonction radiale. Puis on initialise une matrice avec des zéros qui sera la base de notre monde généré. On la remplit finalement avec la formule de la fonction radiale ($g(x) = k\sum i=1 \omega i.\varphi(|x-xi|)$). Tout cela se fait dans le RadialBasisFunction.cpp.
Puis dans le fichier scene.cpp, on construit les cubes du monde avec la fonction qui génère le monde que l'on a codé avant.

e) Ajout de lumières

La lumière ambiante et ponctuelle

"Et la lumière fut ! déclara Toto des étoiles plein les yeux.

Nous avons implémenté deux types de lumières : une lumière ambiante et une lumière ponctuelle.
Peu importe le type de lumière que l'on choisit d'ajouter à notre scène, il est possible dans les deux cas d'en choisir sa position sur les trois axes x, y et z et de la supprimer.
Nous avons paramétré deux ambiances lumineuses par défaut : une ambiance de jour avec la lumière ambiante faisant référence au soleil et une ambiance de nuit faisant référence quant à elle à la lumière d'un réverbère. Il est possible de passer d'une ambiance à l'autre en cliquant sur les boutons correspondants.



La lumière additionnelle

"Et si l'on faisait briller un soleil et un lampadaire en même temps !" s'exclame Toto.

Quelle bonne idée Toto ! Nous l'avons eu avant toi ! En effet, pour se faire, nous avons fait rentrer un nouveau paramètre en jeu : **l'intensité lumineuse maximale**.

Nous faisions varier l'intensité lumineuse entre 0 et 1 pour nos deux types de lumières. Mais un problème se posait alors à nous : si la valeur dépassait 1, alors la valeur n'était plus reconnue. Et là c'est le drame ! Pour répondre à cette contrainte, nous avons alors utiliser la fonction min. Ainsi, si l'addition des deux intensité est inférieure 1, nous conservons ce résultat mais s'il est supérieur, nous gardons la valeur maximale de notre intensité qui est 1. C'est bien connu : deux lumières valent mieux qu'une !

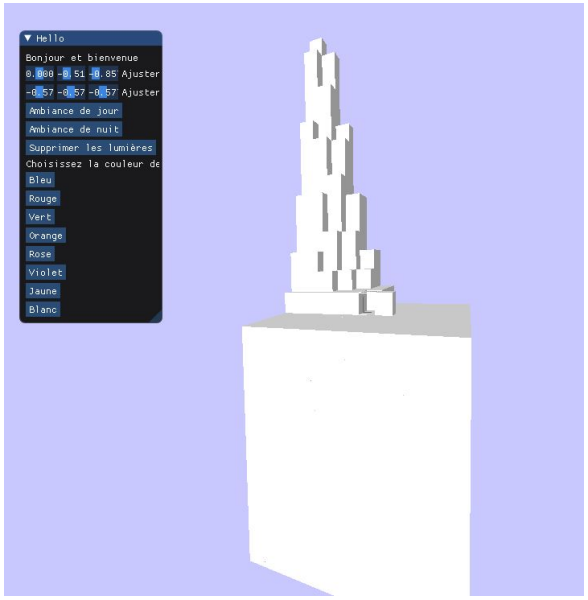
f) Fonctionnalités additionnelles

Nous avions prévu d'ajouter deux fonctionnalités supplémentaires : le texture des blocs et les outils de paintings. Ces deux fonctionnalités nous ont plu parce qu'elles permettaient un rendu visuel plus agréable pour nos utilisateurs et une plus grande créativité (pourquoi pas faire un arbre avec une texture bois ou la mer avec la couleur bleue du painting).
Nous n'avons malheureusement pas eu assez de temps pour les implémenter, nous avons voulu privilégier les fonctionnalités primaires essentielles. Nous aurions aimé avoir le temps de les faire mais notre éditeur est déjà bien complet, nous en sommes quand même satisfaites.

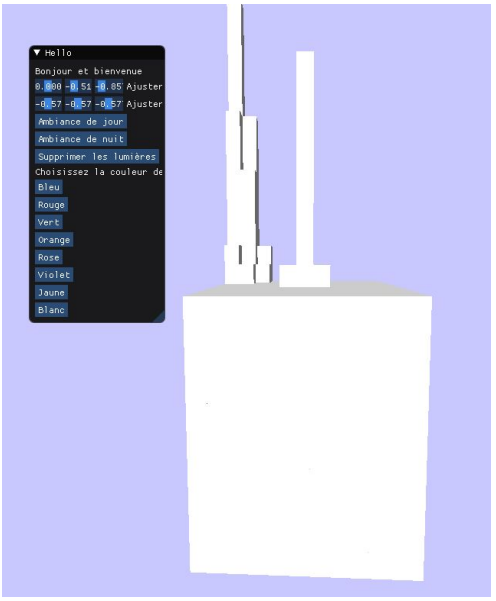
IV. RÉSULTATS DE LA GÉNÉRATION PROCÉDURALE

Nous avons testé plusieurs génération procédurale. En effet, les résultats ont été très différents voire étonnants. Jugez par vous-même :

- **GAUSSIENNE** ($\varphi(d) = \exp(-Ed^2)$)

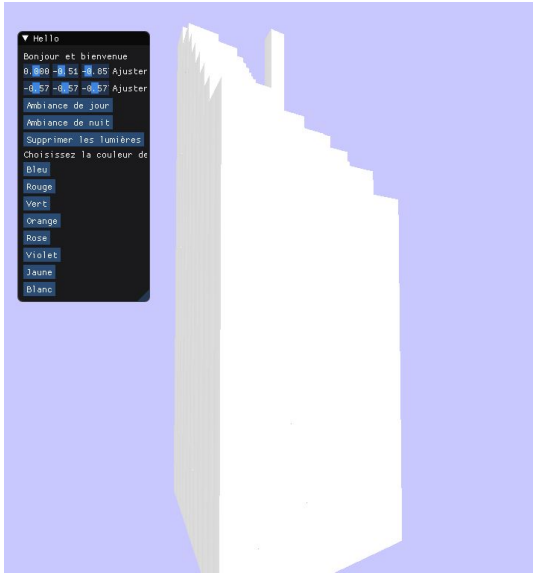


(avec epsilon = 0.1)

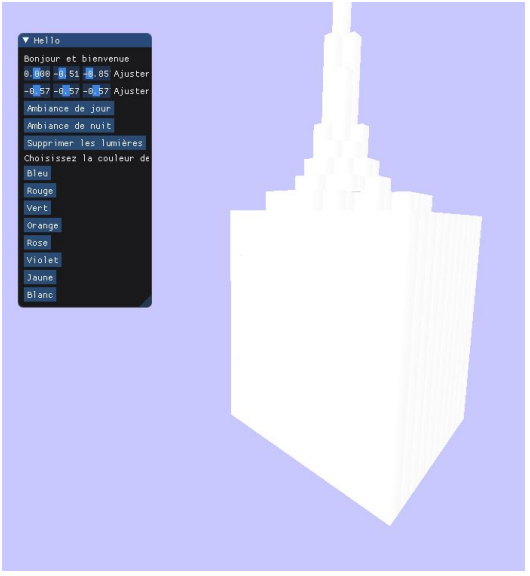


(avec epsilon = 0.5)

- **INVERSE QUADRATIQUE** ($\varphi(d) = 1/(1+(Ed)^2)$)

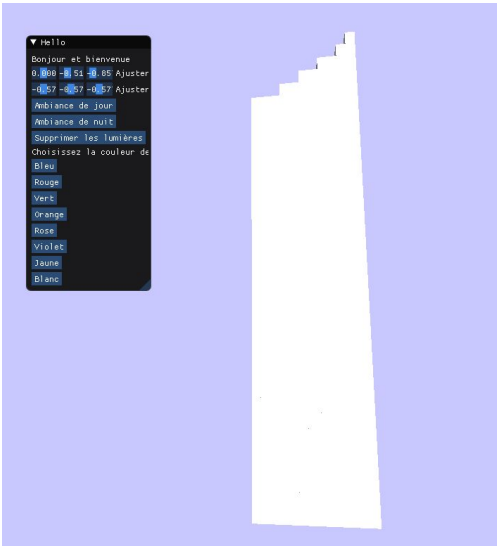


(avec epsilon = 0.1)

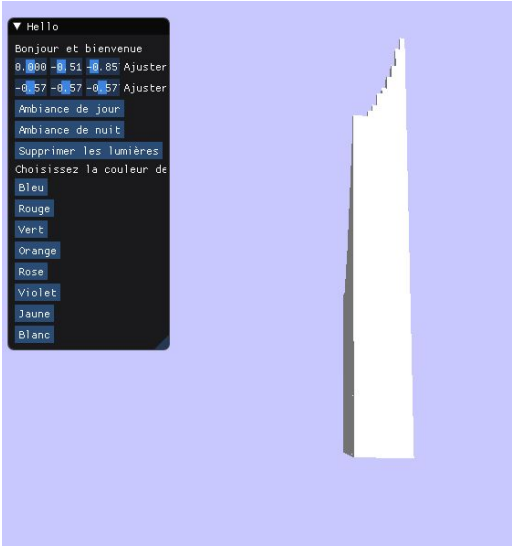


(avec epsilon = 0.7)

- **MULTIQUADRATIQUE** (φ(d) =(1+(*E*d)²)^(½))



(avec epsilon = 0.1)



(avec epsilon = 0.6)

V. DIFFICULTÉS RENCONTRÉES

a) Affichage des cubes

La première difficulté que nous avons rencontré fut l'affichage du cube. En effet, au début nous codions grossièrement dans notre main (càd le *template.cpp*) pour déjà voir si notre code était correct et permettait l'affichage. Notre cube s'affichait sans problème.
Mais dès que nous "rangions" le code du cube dans une classe Cube avec des fonctions dans notre *cube.hpp* plus rien ne s'affichait. Aucune erreur n'était pourtant à déplorer ni aucune fuite de mémoire.
Nous avons alors essayé d'autres méthodes d'affichage (au moins trois différentes), pourtant dès que nous devions le déplacer dans le cube.cpp et cube.hpp, rien ne s'affichait.
Nous avons été bloqué deux jours entiers avec ce problème. Puis nous avons demandé de l'aide à nos camarades (Jules) et ensemble nous avons décelé le problème qui venait du code de nos shaders mais aussi de conflits dans le main.

b) Caméra FreeFly

Pour la caméra nous étions d'abord parti sur une caméra classique avec une simple rotation de l'objet sur lui-même. Mais ce n'était pas très satisfaisant, puisqu'une fois les cubes implémentés et la possibilités d'en ajouter, le monde devenait plus complexe. Nous avions envie de pouvoir nous déplacer librement avec une rotation sur tous les axes.
Nous avons donc implémenté la FreeFLy Caméra mais là aussi nous avons perdu pas mal de temps à cause d'oublis et d'erreurs.
Finalement nous avons réussi à l'implémenter et cela fut très satisfaisant de pouvoir se déplacer librement dans notre monde.

c) Librairie

"Le plus dur c'est de commencer" relativise Toto.

La documentation sur la librairie était très fournie. Il nous a paru difficile au départ de s'y retrouver entre tous ces fichiers. Nous avons pour résoudre ce problème regarder des exemples présents dans la documentation et des tutoriels sur YouTube. Finalement, le plus périlleux a été de l'implémenter car une fois cette

dernière mise en place nous avons trouvé la compréhension de cette dernière facilement et les nombreux exemples nous ont permis de facilement adapter la bibliothèque et donc notre menu et ses boutons à nos attentes.

d) Mac /VS/ Ubuntu

"C'est l'histoire d'un pingouin croquant dans une pomme" ricane Toto.

Nous avons eu des embêtements avec non pas le code mais l'outil de programmation même : nos ordinateurs. En effet, nous avons codé sur nos ordinateurs personnels (puisque la faculté est fermée pendant les vacances et que nous habitons toutes les deux dans le Sud). Mais problème : Cyrielle code sur Mac et Océane sur Linux (Ubuntu).
Le git était donc compliqué pour nous puisque Cyrielle devait avoir un cmake et des lignes particuliers dans le code qui disparaissaient si je mettais mes fichiers.
Nous avons alors pour résoudre ce problème utiliser un gitignore. Les fichiers connaissant des différences dues à nos systèmes d'exploitation n'étaient alors plus pris en charge par GitHub. Quel bonheur : cela nous a bien simplifié les choses par la suite.

VI. PARTIES INDIVIDUELLES SUR LE RESSENTI DU PROJET

a) Cyrielle

Le projet m'a apporté beaucoup d'inquiétude au départ. En effet, j'avais la mauvaise impression de ne pas savoir par où commencer car le projet me paraissait très conséquent. Après avoir relu le sujet plusieurs fois, puis l'avoir découpé avec Océane, cela m'a rassuré car j'ai pu fixer dans mon esprit les différents points à accomplir.
Par la suite, la semaine libérée a été la bienvenue. En effet, nombreux sont nos projets et chargées sont nos semaines alors ces journées nous ont permis de nous concentrer sur ce devoir sans être interrompues. Nous avons à ce moment là bien avancé sur le projet. L'entraide entre camarades à l'université nous a permis de gagner du temps, de comprendre plus facilement et d'être dans un contexte motivant pour la réalisation du projet.
Les vacances de Noël arrivant à grands pas, il a été difficile de lier les fêtes, les projets des autres matières et les révisions de partiels. Nous avons alors préféré rendre une version avec des méthodes opérationnelles et claires plutôt qu'un projet complet mais fait à la hâte.
J'aurais aimé qu'une partie graphique soit compatible avec le projet de programmation. Cela aurait permis une variété de possibilités et de scénarios et aurait ainsi pu refléter notre côté créatif. Néanmoins, le projet m'a plu car il permettait d'expérimenter en temps réel et de réfléchir à comment est conçu un logiciel 3D et ainsi de mieux comprendre comment l'utiliser.

b) Océane

Ce projet me faisait peur au premier abord. Il paraissait assez insurmontable au vue de mon niveau et des exigences demandées.
Les débuts ont été assez lents parce que je devais me mettre dans le bain et que l'on avait beaucoup de projets scolaires en parallèle. Mais une fois le projet commencé, je me suis vite sentie à l'aise et cela fut même une joie dès que l'on réussissait à implémenter une fonctionnalité.
Nous avons avec Cyrielle misé sur la répartition équitable dès le début du projet sauf si nous jugions cela très compliqué alors nous nous mettions toutes les deux sur le problème. Je me suis occupée de la création du cube, de la sculpture du terrain, de la caméra et du curseur.

Le projet était toutefois consistant et les difficultés rencontrées (surtout Mac/Ubuntu) nous ont beaucoup ralenti malgré nos efforts. Le projet étant fonctionnel et agréable à utiliser, nous somme contentes d'avoir privilégier la qualité à la quantité.

La partie que j'ai le moins appréciée fut la partie avant l'affichage c'est-à-dire la mise en place du projet avec les deux jours bloqués sur l'affichage du cube et de la librairie. C'était assez décourageant, nous avions l'impression de ne plus avancer. En plus de cela nous n'avions aucun visuel. Heureusement grâce à des efforts et de l'entraide nous avons surpassé cette mauvaise passe.
La partie que j'ai préféré réaliser est le moment où les premier éléments se sont affichées : le cube, la couleur, la lumière, le curseur… C'est satisfaisant de ne pas coder à l'aveugle et de voir le résultat réel et dynamique. Nous avions l'impression d'être libre dans l'espace de la fenêtre.

VII. CONCLUSION

En conclusion, ce projet a été un vrai challenge pour nous. Nous nous sommes beaucoup investie dedans et nous avons apprécié la création de notre Word Imaker pour notre ami Toto. Le rendu final n'est pas complet par manque de temps, nous restons tout de même sur une image positive du projet et fières de ce que nous avons réussi à implémenter. Il nous a permis de mieux comprendre le langage C++ et la logique algorithmique qui est une approche très utile dans le monde professionnel.