ALGORITHMIQUE

SYNTHÈSE D'IMAGES

DÉSERT'IMAC

Maxime DOYEN Quentin MESQUITA Yvan SMORAG

I. Le projet

a. Présentation et rendu général

ALGORITHMIQUE	
Création du QuadTree	
Dessin du terrain	
Frustum Culling	
LOD	
SYNTHÈSE D'IMAGES	
Chargement du terrain	
Coloration terrain selon hauteur	
Affichage de la Végétation	
Déplacement Caméra	
Caméra fixe au dessus du sol	
SkyBox	
Illumination de Lambert	
Visualisation du QuadTree	
Réparation des cracks	

Voilà ci dessus une vision globale du projet, ce qui a été fait et ce qui n'a pas été fait. Je reviendrai sur plusieurs points qui méritent plus

d’explications avant un retour général sur le déroulé du projet au niveau organisation et les difficultés rencontrées.

b. Implémentation du quadTree

Pour ce qui est du QuadTree, nous avons choisi une implémentation qui nous paraissait cohérente : nous avons une structure qui se compose de quatre “nodes”, à savoir un découpage de l’image en quatre parties. Dans cette structure nous avons aussi quatre points qui servent à délimiter les quatres coins de l’image (et qui devaient servir au LOD même si nous n’avons pas pu l’implémenter cf supra). Il y a parfois quelques bugs de chargement (notamment au niveau du frustum et des coins de la map) si nous ne chargeons pas une map carré donc il serait préférable de travailler avec ce type de heightmap (nous pensons que cela vient du fait que lorsque la map n’est pas une puissance de 2, l’arbre n’est pas équilibré ce qui crée des problèmes de calcul d’intersection).

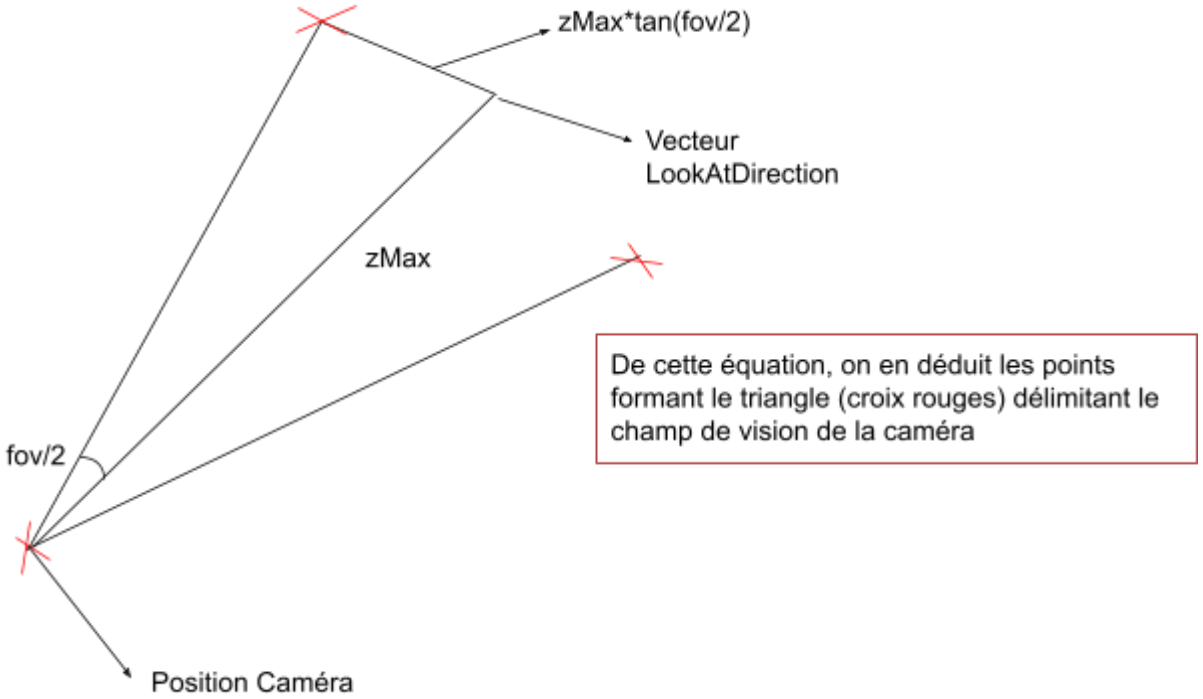
c. Gestion de la caméra et billboard

En ce qui concerne la caméra, nous avons beaucoup navigué à l’aveuglette. Nous avons mis longtemps à comprendre les coordonnées sphériques, et finalement nous avons implémenté quelque chose un petit peu à notre sauce, avec un vecteur LookAt Direction dépendant de la position de la caméra, qu’il est possible de déplacer sur les trois axes (voir commande en annexe). Dans notre cas, nous avons choisi de mettre l’axe Z comme axe vertical, et de placer le terrain sur l’axe OXY. Cette gestion un petit peu personnelle de la caméra, nous a cependant

posé quelques problèmes notamment au niveau du billboard. En effet en passant par des coordonnées non sphériques nous avons eu du mal à récupérer l'angle horizontale de la caméra, ce qui fait que nous avons dû renoncer à avoir des billboards qui s'orientaient par rapport à la caméra : ils se placent seulement de manière aléatoire sur la map.

d. Frustum culling et LOD

Pour le *frustum culling*, nous avons utilisé de la trigonométrie afin de déterminer le triangle de la caméra en projection sur le plan OXY. Nous avons délimité le triangle de la caméra, et dessinons uniquement les points qui sont en intersection avec le triangle de la caméra ou dedans. Le frustum n'a donc pas forcément une image de triangle parfait, puisque pour dessiner une feuille du quadTree il suffit qu'un seul de ces points soit dans le triangle de la caméra ce qui permet d'avoir globalement un frustum culling qui marche même s'il pourrait être amélioré afin de former un triangle bien propre (même si physiquement l'oeil humain fait le flou sur les bords de sa vision, donc avec un angle de vision suffisamment grand et un écran suffisamment large, on pourrait "masquer ce problème").



Pour le LOD, nous n'avons pas réussi à l'implémenter comme nous le désirions. Pour éviter les bugs et nous concentrer sur le reste, nous avons finalement décidé de ne plus l'implémenter. En revanche, sans LOD, l'affichage quadTree donne un résultat "contrasté".

e. Texture

Pour avoir une identité visuelle cohérente, nous avons décidé de donner un aspect "plage" à notre projet. Nous avons pris des textures de ciel bleu un peu paradisiaque, une texture sableuse et des palmiers comme décor. A voir si dans le futur nous pourrons rajouter de l'eau, des oiseaux animés, et pourquoi pas une véritable oasis !

II. Organisation et déroulé du projet

a. Communication du projet

Pour la communication, nous nous sommes réunis plusieurs fois durant le mois de mai pour fixer des deadlines et se répartir le travail afin de ne pas être trop pressé par le temps. Nous avions un repository GIT sur lequel nous pushions chaque nouvelle fonctionnalité pour ne pas avoir trop de conflits de merge et un code commun cohérent. Il nous est arrivé de nous retrouver sur Discord pour coder quelques points plus compliqués ensemble (ou demander de l'aide à d'autres personnes), et faire des points assez régulierement ou nous entraider pour certaines parties du code un peu plus complexes (frustum culling ou le quadTree par exemple.

b. Organisation du projet

Pour éviter d'avoir des fonctions trop longues et trop de codes à lire, nous avons essayé d'avoir une architecture logicielle la plus propre possible : nous avons donné des noms les plus explicites possibles à nos différentes variables et fonctions, commenté au maximum, et séparé chaque module du code en fichier (via un makefile qui compilait automatiquement tous nos fichiers d'extension .cpp) afin d'avoir une relecture pas trop ardue du code d'autrui. Nous avons donc :

⇒ Un fichier *geometry.cpp* contenant les fonctions de calcul vectoriels, (scalaire, addition de vecteurs) et d'intersection de droite/triangle

⇒ *LoadSDL.cpp* qui contient des fonctions de loading de la SDL, des fenêtres, textures etc….

⇒ *getFile.cpp* qui charge la heightMap et les différents paramètres selon le fichier fourni en argument du programme

⇒ *Light.cpp* qui charge et affiche la skyBox

⇒ *Camera.cpp* qui contient les fonctions de frustum Culling et de calcul du champ de la caméra

⇒ QuadTree.cpp qui implémente toute les fonctions relatives à la manipulation et l'initialisation du quadTree

⇒ *terrain.cpp* qui dessine le terrain et les billboards selon les différents modes demandés (dessin du quadTree, en mode filaire etc).

⇒ Le fichier main.cpp qui rassemble toute les parties en un tout cohérent

⇒ Tous les fichiers .cpp sont doublés (à l'exception de main) d'un fichier header et d'un makefile pour tout compiler sans problème. Nous avons aussi utilisé la bibliothèque SDL_IMAGE, pour charger les différentes images de textures.

Pour la répartition des tâches, Yvan s'est occupé du quadTree, Quentin de la SkyBox, et Maxime du billboard. Pour les fonctions de terrain, de gestion de la caméra nous avons tous participé et codé différents bouts pour s'assurer que tout le monde comprenne bien le déroulé des algorithmes et puisse travailler de son côté en utilisant le travail des autres.

ANNEXE : Répartition des touches :

Gestion de la position :

- **Z, Q** ⇒ Se déplacer sur l'axe x
- **S, D** ⇒ Se déplacer sur l'axe Y
- **[espace], x** ⇒ Se déplacer sur l'axe Z

Gestion de la caméra :

- **[UP], [DOWN]** ⇒ Bouger la caméra sur l'axe Z
- **[Left], [Right]** ⇒ Bouger la caméra sur l'axe X
- **L, M** ⇒ Bouger la caméra sur l'axe Y

Différents modes de la caméra/terrain :

- **F** ⇒ Passer en mode fil de fer
- **G** ⇒ Activer/Désactiver le Frustum Culling
- **I** ⇒ Mode caméra au dessus du sol
- **C** ⇒ Afficher/Désactiver le triangle de la caméra
- **N** ⇒ Afficher/Désactiver le mode QuadTree