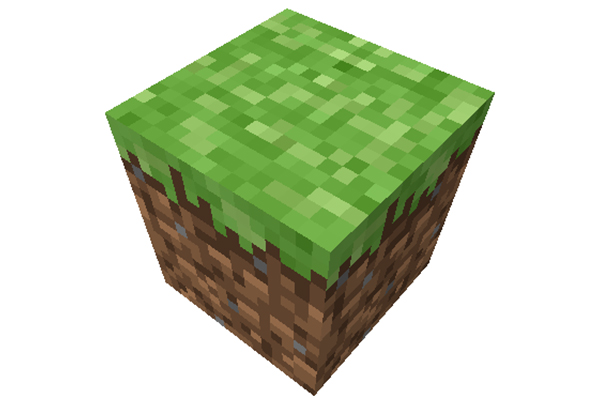
Rapport du Projet1

Réalité Virtuelle

**Minecraft simplifié**

Etudiants : Wei ZHOU & Ye ZHOU

Encadrant : Jean-Marie Normand

Le 27 janvier 2016

Index

I. Sommaire ......................................................................................... 3

II. Modules ........................................................................................... 4

III. Problèmes & Solutions .................................................................. 16

IV. Désavantages et idées à réaliser.................................................... 20

V. Résultat final ................................................................................. 21

VI. Conclusion .................................................................................... 25

VII. Bibliothèque ................................................................................... 25

1. **Sommaire**

Tout d’abord, notre sujet est < **Jeu vidéo OpenGL/OSG** >, ça va dire que nous devons écirie un jeu vidéo avec le langage OpenGL ou OSG, et il nous fait plaisir que nous pouvons définir quel jeu vidéo nous allons écrire.

Donc nous choisissions le jeu vidéo des cubes très connu –**Minecraft** comme notre thème du projet. Et pour écrire ce projet, nous avons beaucoup utilisé le bibliothèque **GLUT** (Open**GL** **U**tility **T**oolkit).

Pour le raison de temps, nous n’avons pas réalisé tous les fonctions dans le Minecraft, seulement une partie des fonctions, par exemple : se déplacer, enlever un cube, déposer un cube, créer des édifices avec des cubes différents......

Nous allons les présenter en détail au texte suivant.

Nous ne pouvons pas nier qu’il existe quelque désavantages dans notre projet, mais nous nous avons fait du mieux et bien fait attention, et nous espérons que notre projet vous fera plaisir ☺

1. **Modules**

Nous avons écrit beaucoup de modules dans notre projet, nous croyons que c’est plus claire et plus facile pour lire et comprendre notre structure et les fonctions.

1. **Eléments du monde : Cubes et plantes**

**Introduction** : le monde virtuel est composé par les cubes et plants.

**Classe :** Cube, Plants

**Location :**

* cube.h cube.cpp
* plants.h plants.cpp

**Membres essentiels :**

* Les coordonnées de cube et plant

float positionX, positionY, positionZ;

* Le type de cube et plant

int type;

* Un identificateur pour juger si un objet est sélectionné

int chosen;

**Methods Majors:**

* obtenir le texture a partir de un type :

gettexture()

* affichage de un cube/plante :

afficheCube()

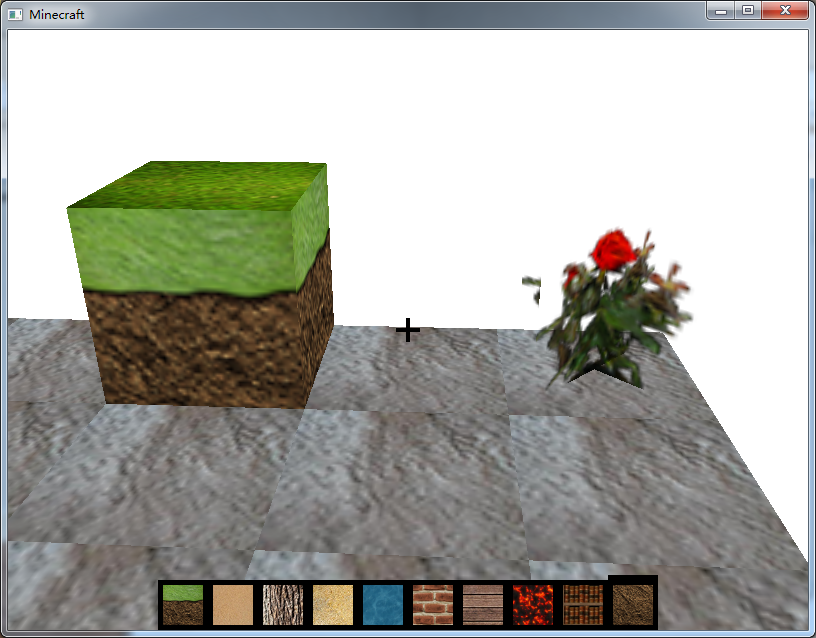
* obtenir les propriétés:

getpositionX()

getpositionY()

getpositionZ()

gettype()



**Il est un cube et un plante**

1. **Le monde virtuel : world**

**Introduction** : Comme Minecraft est un jeu bac à sable (Sandtable game). On a besoin tout abord un monde virtuel.

**Classe :** World

**Location** : world.h world.cpp

**Membres Majors :**:

* Enregitre les cubes et plants

map<int,Cube> cubes;

map<int,Plants> plants;

* Un identificateur pour juger si un objet est sélectionné et les coordonnées

int chosenx; int choseny; int chosenz;

int chosen;

* Le cube ou plante avant le cube choix

int frontx; int fronty; int frontz;

* Le point de vue

View viewer;

**Methods Majors:**

* Ajouter et suprimer les cubes and plants:

World ::addCube()

World ::addplants()

World ::deleteCube()

World ::deleteplants()

* Affichage de le monde virtuel:

World ::generation()

World ::afficheworld()

* La collision

World ::collision()

1. **Le point de vue : viewer**

**Introduction** : le caméra. En fait c’est un personne virtuel dans le monde.

**Classe :** View

**Location** :

* world.h world.cpp
* View.h view.cpp
* Mouse.h
* Keyboard.h

**Membres Majors :**

* Les coordonnées

float mypositionX, mypositionY, mypositionZ;

* La direction de point de vue

float object, object, objectZ;

* Le vitesse

float speedx, speedy, speedz;

* L’ accélération

float accelerx, accelery, accelerz;

**Methods Majors:**

* Controôle la deplacement du viewer avec le souris.

Mouse::mousemove()

* Déplacement : avancer, reculer, tourner à gauche, tourner à droité, saut à haut.

Keyboard::keyboardmovement()

* Choice un objet : On tracer une ligne droite a partir de la position de viewer et tracer par le objet direction.

World:: viewerchoose()

* Clique evenement : cliquer gauche pour destuire un cube et cliquer droite pour mettre un cube.

Mouse:: leftclick()

Mouse::rightclick()

World:: vieweraddCube()

World:: vieweraddplants()

World:: viewerdelete()



Dans cette image, cette case est cochée. Il y a une bordure rouge autour

1. **Les autres modules**
2. **SOIL**

**Introduction:** Parce que nous avons besoin d'utiliser un texture transparent, ce qui faut l'utilisation image **png**. Mais le format png est difficile à lire, donc nous utilisons un bibliothèques tierces: soil

**Location :** main.cpp

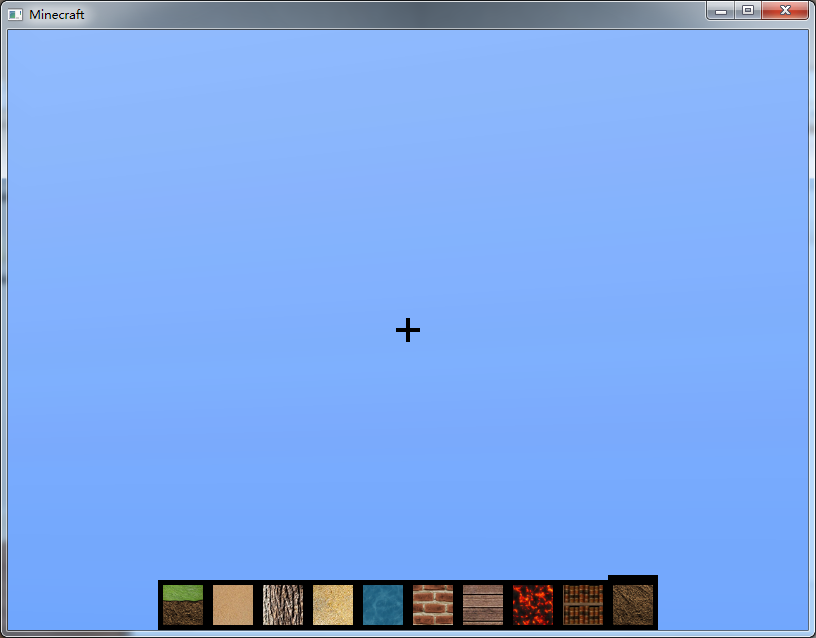
**Methods Majors:** SOIL\_load\_image ()

1. **2D GUI**

**Introduction :** Le 2D GUI est aussi important pour interaction avec le personne.

**Location :** GUI.h

**Methods Majors:** drawGUI()



La petite croix dans le milieu indique que le curseur. La barre d'outils ci-dessous montre les cubes. Il sera marqué sur le cube sélectionné.

1. **Enregistre et rapporter de un monde**

**Introduction :** ce module nous permette d’enregistrer et rapporter un monde a partir d’un fichier. Le fichier est save.txt.

**Location :** world.h world.cpp

**Methods Majors:**

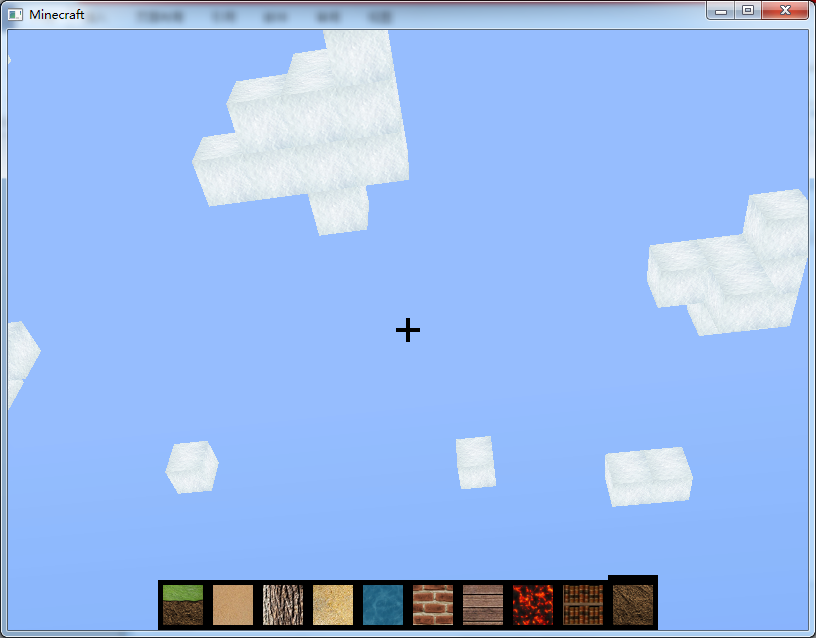
* void savetheworld();
* void readtheworld();

1. **skybox**

**Introduction :** Le skybox est un boite avec texture qui presente le ciel.

**Location :** skybox.h

**Methods Majors:** skybox()



C’est le skybox et quelques nuages

1. **font**

**Introduction :** Le skybox est un boite avec texture qui presente le ciel.

**Location :** skybox.h

**Methods Majors:**

* Font::BuildFont ()
* Font::glPrint()
* Font::KillFont()

Finalment on n’utilise pas les fonts.

1. **Problèmes & Solutions**
2. **La combination entre 2D et 3D**

Ce n’est pas facile de combiner le 2d GUI et 3d monde virtuel ensemble avec glut. Nous avons fait beaucoup de recherche sur l’internet. Mais nous n’avons pas trouvé la méthode. Merci à monsieur Normand, il nous a conseillé de changer le mode de matrix avec :

glMatrixMode(GL\_PROJECTION);

et

glMatrixMode(GL\_MODELVIEW);

Nous l’avons trouvé très utile, cette méthode, mais il y a encore un problème : apres avoir affiché le 2D GUI, le 3D monde est devenu un image 2D ! Ensuite nous avons essayé beaucoup de méthodes et finalment, nous avons trouvé la solution :

glDisable(GL\_DEPTH\_TEST);

glDisable(GL\_BLEND);

glMatrixMode(GL\_PROJECTION);

glPushMatrix();

glLoadIdentity();

gluOrtho2D(0.0, 800, 0.0, 600);

glMatrixMode(GL\_MODELVIEW);

glPushMatrix();

glLoadIdentity();

// affichage de 2d interface.

glMatrixMode(GL\_PROJECTION);

glPopMatrix();

glMatrixMode(GL\_MODELVIEW);

glPopMatrix();

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_BLEND);// you enable blending function

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

L’étape clé est diable le GL\_DEPTH\_TEST avant afficher le 2d et l’enable après.

1. **L’affichage des cubes transparents**

Dans les version archaique nous avons utilisé un image BMP comme les textures. Quand nous avons écrit les plants, il faut utiliser transparence pour les afficher mieux. Mais il n’y a pas l’alpha dans BMP. Donc nous l’avons changé avec PNG.

Malheureusement, c’est plus compliqué d’apporter un image PNG. Ce format est deficile de lire.

Finalment nous avons utilisé un third-part-library : SOIL. Après avoir complié le code original et l’ajouté dans le projet, on peut rapporter un PNG avec SOIL\_load\_image() et glTexImage2D()

Mais après nous avons fait ça, le 2D GUI disparu !! Nous ne savons pas le raison et nous avons fait beaucoup de cherches sur internet. Finalement nous avons ajouté glDisable(GL\_BLEND); avant l’affichage de 2d GUI et glEnable(GL\_BLEND); Ca marche en fin!



Ceci est un matériau transparent de un plante, nous pouvons voir qu'il est partiellement transparent

1. **Problèmes de deplacement**

Parce que le point de vue du persenne changera la définition de "front", pour lui est difficile de juger la direction de marcher. Ici je utilise la méthode de vecteur. On utilise PositionXYZ : la position du jeu de caractères. ObjectXYZ : la direction face à. Dans le processus de déplacement de vue de point, avec une traduction de vecteur et de rotation pour contrôler la direction de l'orienté. Par exemple , la rotation de vecteur:

float x = world.viewer.objectX;

float z = world.viewer.objectZ;

world.viewer.objectX = cos(ang) \* x + sin(ang) \* z;

world.viewer.objectZ = -sin(ang) \* x + cos(ang) \* z;

Aussi, Je utilise les lois de la physique pour contrôler le mouvement des personnages a sa vitesse et d'accélération :

if(speedx != 0 || speedz != 0)

{

accelerx = -0.5 \* speedx;

accelerz = -0.5 \* speedz;

}

mypositionX += speedx;

mypositionY += speedy;

mypositionZ += speedz;

speedx += accelerx;

speedy += accelery;

speedz += accelerz;

J’ai essayé de contrôler directement les appuis de déplacement. Mais opengl ne peut identifier qu’un seul événement. Plusieurs clés pressé dans la meme temp est invalide. Donc, j’ai créé une table d'état pour enregistrer les appuis qui sont en cours de presser. Cela peut gérer plusieurs touches pressées :

int keys[200];

void setkeydown(char key){keys[key] = 1;return;}

void setkeyup(char key){keys[key] = 0;return;}

1. **Lentement si trop de cubes**

Quand nous somme en train de fonder mon monde virtuel. Il est bloqué tout à coup. Nous avons trouvé que le problème est parce que nous avons mis trop de cubes et plants dedans, et l’espace de dépôt n’a pas assez et la réaction de ce jeu est de plus en plus lent. Donc nous commerçions à penser utiliser un structure de donners differents.

Finalement nous avons trouvé le recipient : Map dans STL de C++. C’est un structure à partie de le mot clé et le content. Ensuite je utilise a hash method alors je peux trouve un Cube ou un Plant plus vite avec le XYZ position. Le hash algorithsm est :

int num = 1000000\*(MAX+xx) + 1000\*(MAX+yy) + (MAX+zz);

plants.insert(map<int, Plants> :: value\_type(num, p));

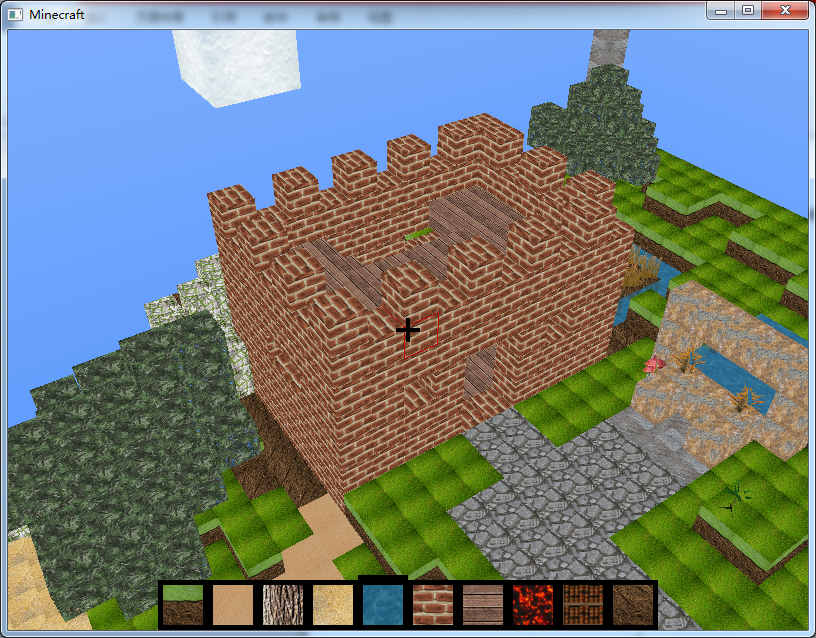
Par conséquent, le jeu peut être un mode de fonctionnement normal

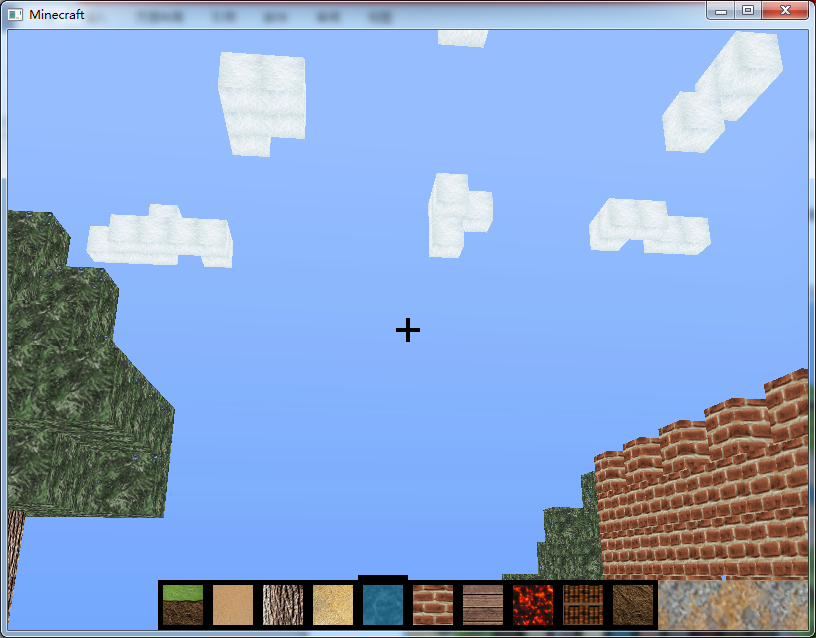
1. **Résultat final**

Enfin, nous avons construit notre beau monde dans le jeu!

Ce qui suit est une vue à vol d'oiseau







**les contrôles :**

1. Appuyez sur W, S, A, D, à aller de l'avant, arrière, gauche, allez à droite. Vous pouvez appuyer ensemble les deux cisailles.
2. Appuyez sur la barre espace pour sauter, a fait pression sur l'espace peut contenir à voler
3. Déplacez la souris pour changer l'angle de vue. Appuyez sur le bouton gauche de la souris pour détruire un cube, appuyez sur le bouton droit de la souris pour placer un cube
4. Appuyez sur les numéros 0-9 pour changer le cube sélectionné. Vous pouvez appuyer sur + et - pour retourner
5. Appuyez sur Esc pour quitter le jeu, il va enregistrer vos modifications, et la prochaine fois que vous ouvrez le jeu conservera ces changements. Si vous ne voulez pas enregistrer, vous pouvez quitter en appuyant o
6. **Désavantages et idées à réaliser**
7. **Cubes transparents :**

Pour des cubes avec la diaphanéité, par exemple : des fleurs, des herbes ou des feuilles -- différents des cubes normal (sol, pierre, bois...), ils ne sont pas des cubes complets, il y a des epaces transparents, comme on peut voir de ces photos suivantes :



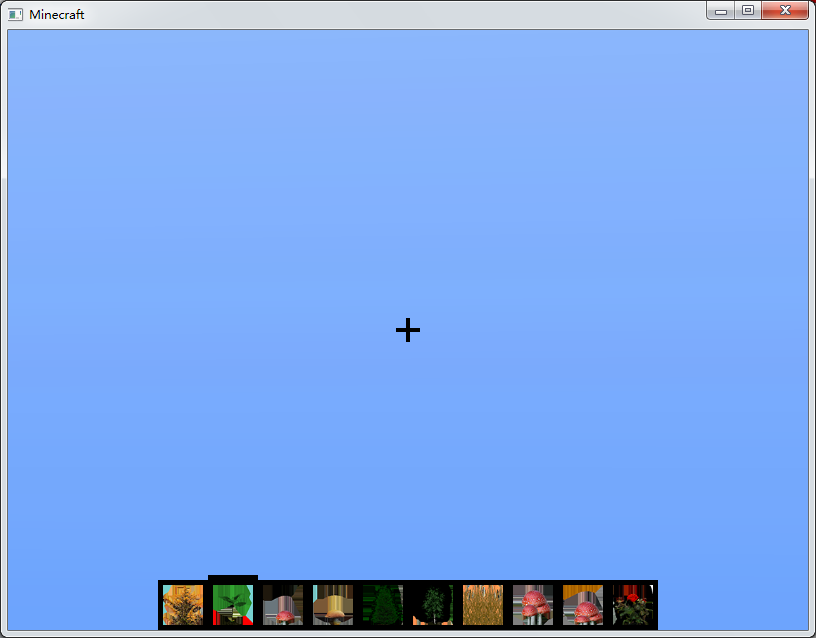
Mais dans notre projet, quand on change son angle visuel, de temps en temps, il existe un petit problème d’affichage :

au centre d’objet, il y a une partie incomplet :

Nous ne savons pas le raison de ce problème, mais nous croyons que c’est le problème de notre moteur

1. **Transparents dans le 2d GUI**

Dans l'interface utilisateur, nous avons un problème étrange, quand nous serons image matérielle imprimée sur le 2D GUI, il y aura des couleurs étranges. Comme l’image dessous :



Ceci est un problème que nous ne pouvons pas résoudre

1. **Trop seul..**

Nous avons généré le monde qui est très beau, mais il y a encore un problème: le joueur est trop seul !! Nous pouvons également ajouter certains animaux et les gens dans le jeu, mais cette fois nous avons peu de temps. Si on peut continuer à faire notre projet, je vais d'abord ajouter les animaux et les humains.

1. **ne peut pas fonctionner sur d'autres OS**

Nos jeux ne peuvent fonctionner que dans Windows. Bien que j’ai tenté d'exécuter en autres OS, ça marche pas. Je ne sais pas comment faire

1. **manque musique**

J’ai essayé d'ajouter de la musique, mais quand je joue de la musique, le jeu a arrêté et en attente de la musique se fait. Je pense que ce besoin d'utiliser la technologie parallèle.

1. **Conclusion**

Après avoir fait ce projet, nous avons mieux compris et maîtrisé OpenGL. De plus, parce que le projet nous permettait beaucoup de liberté, donc

1. **Bibliothèque**
2. [OpenGL] Combiner 2D et 3D :

<https://openclassrooms.com/forum/sujet/opengl-combiner-2d-et-3d-66523>

1. Blending par Learn Opengl

<http://learnopengl.com/#!Advanced-OpenGL/Blending>

1. OpenGL Texture Transparente

<http://www.developpez.net/forums/d415052/c-cpp/cpp/opengl-texture-transparente/>

1. map - C++ Reference

<http://www.cplusplus.com/reference/map/map/>

Enfin, grâce à notre aide de professeur Jean-Marie Normand !