SDL_glSetRelativeKebabMode

Image et Son (mais surtout Image)

CHAMPROY Lilian
DUBROCA Axel

Licence 3 Informatique
Groupe IN601A2
Avril 2016

FIGURE 1 – Tribute to Minecraft, parce que cubes et OpenGL, vous voyez ?

Il m'a semblé nécessaire de faire un préambule, afin de prévenir de ce qui va suivre.

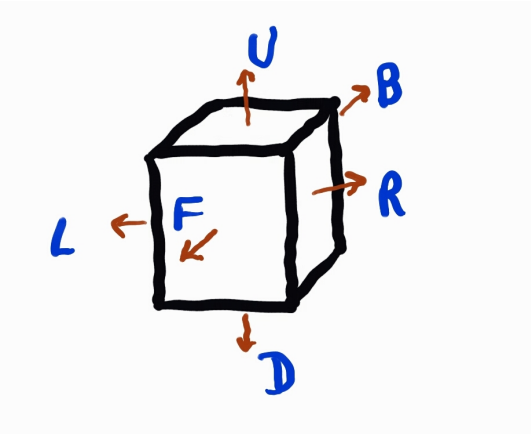
Cet exercice a été réalisé sur une configuration différente, puisqu'il n'a pas été fait sur les ordinateurs du CREMI, pour la simple et bonne raison que c'étaient les vacances. Nous avons passé plus de 4 heures [1] à essayer de faire fonctionner OpenGL sous Windows. Pourquoi avoir tenté sous Windows ? Une des forces d'OpenGL, c'est d'être compatible sous toutes les plateformes. Mais ces 4 heures ont suffi à nous faire abandonner l'idée. Nous avons réussi à faire fonctionner SDL, OpenGL, mais GLEW et GLM, c'était impossible (tout du moins, pas trivial). Nous sommes donc passés sous Linux.

Mais alors que nous pensions que ce serait plus simple - l'installation l'était - il a fallu que les choses soient plus complexes également. Les fichiers sources, et corrections ne compilaient pas. Pourquoi ? Parce que la configuration du CREMI est entièrement personnalisée. Comment, nous n'en savions rien. Quelle solution avons-nous adoptée ? Nous avons tout refait depuis zéro. Nous avons appris à développer avec OpenGL en C++, puis nous avons tout traduit en C, au moins pour tout ce qui était possible. La seule chose que nous ne pouvions pas traduire était GLM, qui est pensée avec le C++. Heureusement, le compilateur étant g++, nous avons pu, ignoblement, insérer des instructions C++ dans nos fichiers sources C. Rien que de le dire, j'en ai envie de vomir. Mais nous n'avions aucun moyen d'outrepasser le namespace glm.

Le résultat que vous avez ici peut donc ne pas compiler sur votre machine. Nous n'en savons rien. Nous mettrons donc des captures d'écran, au moins pour que le rapport se tienne à lui-même. Si vous parcourez le code, vous verrez également que le tout est bien plus massif que ce qui était fait en TP. Pourquoi, je n'en sais rien [2]. Mais nous avons tout de même, en quelques jours, appris l'OpenGL de zéro, tout en traduisant le tout en C. Et ça... On en tire une certaine fierté. Non pas d'un résultat parfait, ni correct, mais de savoir que nous nous sommes au moins approchés du résultat final, malgré tout.

1. Et il n'y a aucune exagération, vraiment, cet exercice a pris, au total, plus de 25 heures à être réalisé.
2. A vrai dire, je pense savoir, puisque nous n'avons assisté à aucun des cours sur OpenGL...

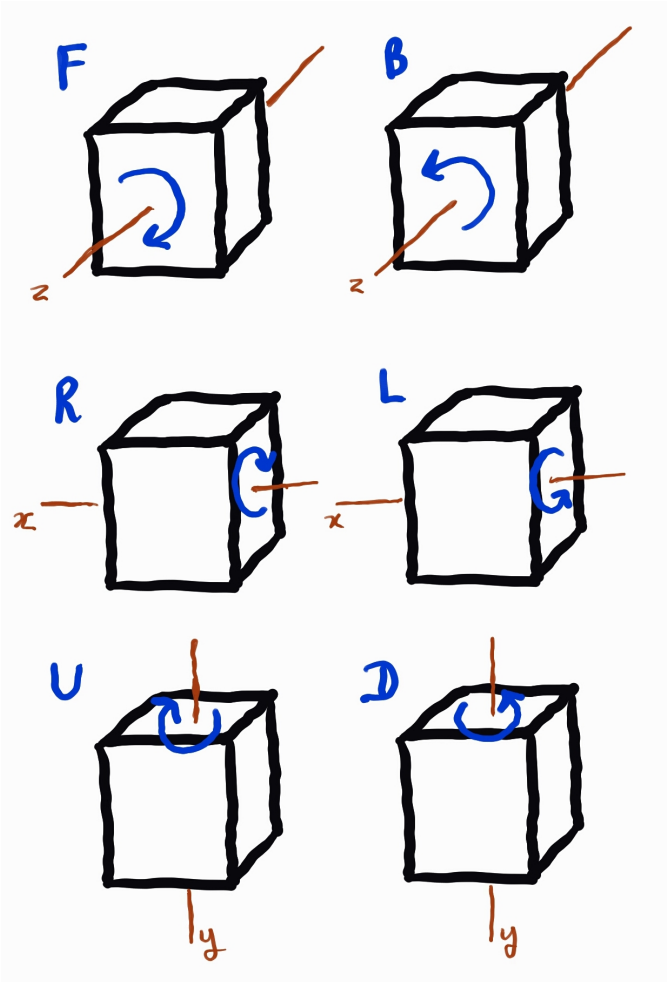
Commençons par une notation pour qualifier les mouvements des cubes. Nous tirons cette notation des Rubik's cubes - c'est ainsi qu'on écrit les algorithmes de résolution du cube. Vous allez comprendre.



(Ne vous inquiétez pas... C'est le fond de l'image qui n'est pas blanc, mais bleuté)

La notation est la suivante : U pour UP, D pour DOWN, F pour FRONT, B pour BACK, L pour LEFT et R pour RIGHT. Simple, non ? Il suffit de considérer que vous voyez, d'une certaine manière, le cube de face.

Du coup, comment décrire les mouvements du cube ? Eh bien, le mouvement "F" indiquera que, si on se place devant la face F, on fait tourner le cube de 90° dans le sens horaire. "B" décrira le mouvement inverse (si on se place devant la face B, et qu'on la fait tourner sans le sens horaire, cela revient à faire tourner le cube dans le sens antihoraire si nous sommes face à F). Voici un schéma récapitulatif :



C'est avec cette notation qu'on décrira les interactions que l'utilisateur aura avec la scène. Évidemment, il s'agit des mouvements par rapport au repère de l'espace, et non pas de la caméra. Si vous vous placez der-

rière les cubes, tout sera inversé ! Voici donc les contrôles (en AZERTY) pour les différents mouvements.

Touche	Cube
O	Right
L	Left
K	Up
M	Down
I	Back
P	Front
SPACE	Grossir
SHIFT	Réduire

Il s'agit d'une caméra libre, configuration standard.

Touche	Caméra
Z	Avancer
S	Reculer
Q	Gauche
D	Droite
Souris	Orienter

Pour le cube, testez les commandes, elles sont plus évidentes que ce qu'il n'y paraît. Nous vous recommandons d'ailleurs, au lancement du programme, de bouger un peu la souris pour vous recentrer. La caméra a un petit soubresaut lors du premier mouvement, qui peut surprendre. La position originale de la caméra (si vous ne bougez pas) est celle de référence pour les mouvements des cubes.

Afin de pouvoir réaliser tout ceci, nous avons suivi un tutoriel pour apprendre rapidement à développer en OpenGL. Et comme nous sommes fainéants, nous avons suivi celui d'Open Classrooms, comme souvent, tout en gardant à l'esprit l'exercice demandé, afin de suivre la voie dont nous avions besoin.

Nous sommes conscients, en comparant notre code aux corrections des TPs, que cette solution est immensément lourde. Nous nous sommes compliqués la tâche, mais aussi, nous avons du laisser tomber l'éclairage et les shaders. Les shaders utilisés ici sont ceux du tutoriel. Nous avons perdu beaucoup de temps à traduire du C++ vers le C, et ce, même partiellement, car nous n'avons pas pu passer toutes les limites (notamment pour GLM).

L'exercice que nous rendons est donc incorrect, et incomplet, mais nous le rendons tout de même, car il contient une partie de ce qui était demandé, en OpenGL, ce qui, en soi, ne déroge pas vraiment aux consignes. Seule la méthode employée diffère.

Dans le cas où le code ne compilerait pas - puisque nous avons une configuration probablement différente - nous avons mis en ligne quelques secondes du programme pour montrer le déplacement de la caméra, et les interactions des cubes. Il s'agit là de montrer au moins quelque chose si jamais le destin serait vraiment contre nous. Le framerate de la vidéo est grandement inférieur à celui du programme (qui tourne à 50 fps). C'est du au fait qu'il a fallu trouver une solution compatible avec Linux pour pouvoir enregistrer la fenêtre... Et c'est loin d'être parfait.

Voici tout de même le lien :
https://youtu.be/pzBXBPMAAmg.

Et si jamais besoin, pour une raison X ou Y, l'adresse du dépôt sur GitHub :
https://github.com/Pyraexyrin/SDL_glSetRelativeKebabMode.git.

Cordialement.