Notes d'intention - DIGGER - C++ & OpenGL

Honorine Rucelle & Louise Januel

Menu de démarrage

Introduction

Dans le cadre de nos cours de C++ avec M. De Smet et de synthèse d'image (OpenGL) avec MM. Biri et Dumazet, nous avons programmé un jeu inspiré de Mr. Do!, Digger ou encore Pacman dans lequel le but est de se déplacer sur une carte pour collecter tous les objets disponibles, avec la possibilité de miner tout en évitant ennemis et pièges.

Nous avons respecté les différentes contraintes dont l'utilisation de Github qui nous a permis de nous organiser et de travailler en duo ou de la librairie glbasimac qui nous aura causé plusieurs problèmes qui seront développés plus bas.

Initialisation

Avant toute chose nous sommes parties du template envoyé sur le serveur Discord par Guillaume à la demande de M. Fouchy. Cela nous a permis de partir sur une base propre, avec un OpenGL opérationnel et une arborescence complète.

Map et items

La première étape a été la génération de la carte en utilisant un algorithme de génération procédurale de type *cellular automata*. La carte est constituée de blocs vides ou pleins (sol et mur) mais également d'obstacles, d'objets et de pièges.

- Les obstacles sont des blocs incassables qui sont mis de manière aléatoire sur la carte, entourés de blocs minables.
- Les objets permettent de gagner la partie si on les ramasse tous (15), ils sont placés aléatoirement également sur des blocs vides.
- Les pièges nous font perdre la partie on vérifie les blocs vides et si

Joueur et ennemis

L'apparition du joueur et des ennemis se fait aléatoirement sur une case vide de la carte.

Le joueur a la possibilité de se déplacer grâce aux touches ZQSD et de miner des blocs de type mur grâce à la touche E. On vérifie les collisions pour qu'il ne puisse se déplacer que sur des cases vides et on gère les déplacements en diagonale. Pour les pièges et les objets, on fait bien attention à vérifier que les 4 coins de l'item répondent s'il y a contact.

Les ennemis eux, nous ont donné plus de fils à retordre. Les collisions étant déjà faites, les espaces dans lesquels ils se déplacent étaient déjà définis, cependant il a fallu utiliser un algorithme de type Breadth First Search ce qui a été plus difficile. Aujourd'hui, les ennemis se déplacent mais prennent parfois du temps lorsque l'un de leur coin entre en contact avec un coin de mur, celui-ci perd énormément en vitesse avant de finalement repartir. L'ennemi, quand il est entouré de mur et que le joueur est à l'extérieur de ce cercle, reste statique jusqu'à ce que les blocs soient minés. Autrement il suit bien notre joueur au travers de la map et lorsqu'il le rattrape, le jeu est considéré comme perdu.

Textures

Les pions sont affichés grâce à des textures en OpenGL qui elles aussi ont été compliquées à gérer. L'objectif initial était de gérer les textures avec des UVs, c'est-à-dire de n'afficher qu'une partie de l'image, ça permet par exemple d'animer les sprites sans changer la texture affichée à chaque fois et donc d'optimiser les performances. Malheureusement nous n'avons jamais réussi cette technique et nous sommes rabattues sur plus simple : le mapping des textures full directement. On définit les quatre coins et on applique l'image sur la partie correspondante. C'est une forme d'UVs mapping simplifiée.

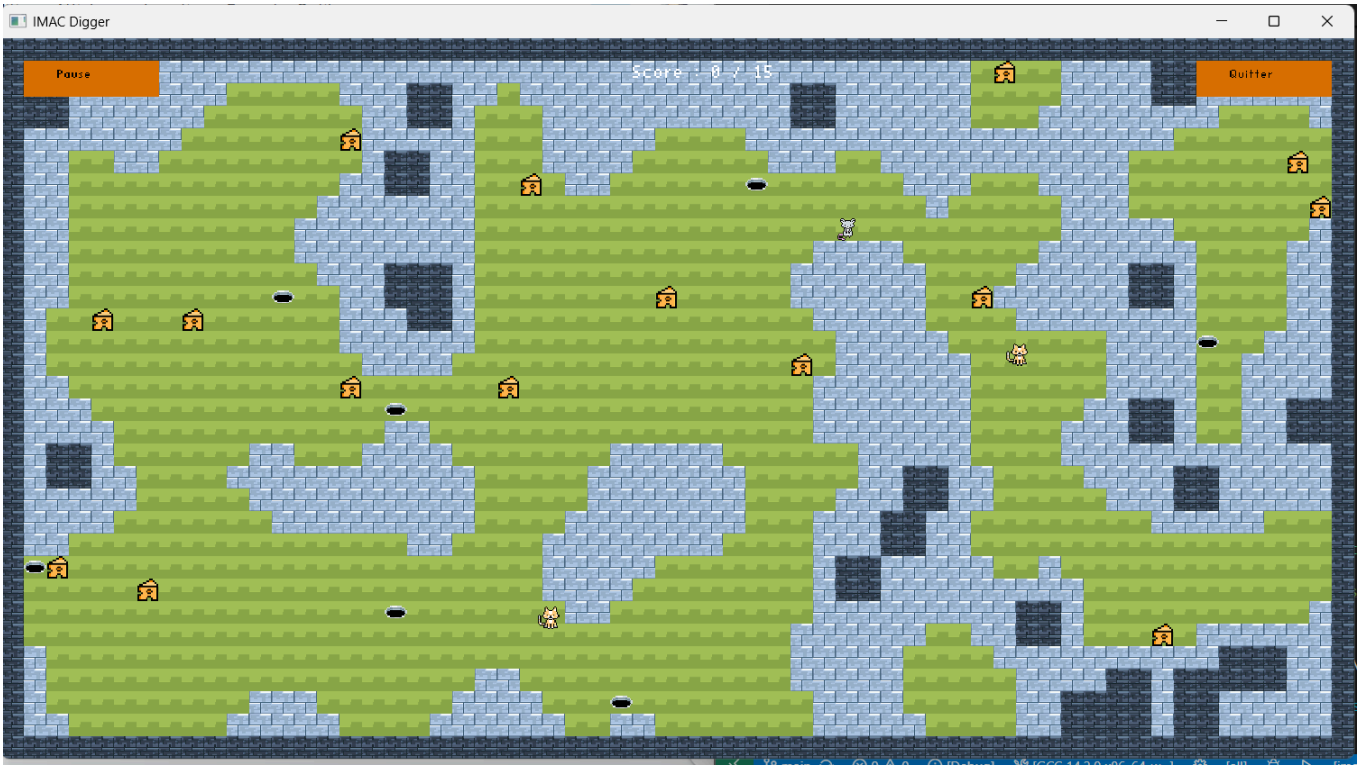
Pour les objets et les pièges, nous n'avons également pas réussi à superposer un fond de sol à un item avec un fond transparent : soit celui-ci était noir soit complètement transparent et laissait apercevoir le fond d'écran de l'ordinateur

directement. Nous avons alors pris le parti d'intégrer la texture sol directement à l'arrière des sprites objets et pièges.

Interface graphique

Le projet comprend une interface graphique gérée par OpenGL. On y retrouve évidemment la carte de jeu avec le joueur, les ennemis, les différents items mais on peut également y retrouver les images de menu et les différents boutons.

Le menu principal est composé d'une image avec deux boutons cliquables responsive, qui nous permettent de quitter ou démarrer le jeu mais le plus intéressant se trouve pendant la partie. La fenêtre de jeu dispose d'un overlay qui contient deux boutons (quitter et pause) et le score. Le bouton pause affiche, à l'image du menu, une photo avec un bouton cliquable. Pour le score, il est géré grâce à stb_easy_font qui nous permet d'utiliser du texte pur directement et s'incrémente à chaque fromage récupéré par la souris. Nous avons également deux écrans de fin : victoire ou défaite.



Interface in game

Conclusion

Avec plus de temps nous aurions aimé améliorer le flowfield pour rendre les ennemis plus fluides et aurions passé plus de temps sur le design qui nous tenait à coeur mais restait secondaire. L'optimisation du code est également un sujet sur lequel nous aurions aimé nous pencher davantage.

Globalement, le projet nous a permis de nous améliorer et de découvrir de nombreuses nouvelles fonctionnalités, notamment sur l'utilisation des structures et l'affichage des textures en OpenGL. (Nous sommes également des pros de Fork désormais.)