LANGUILLE Antoine Groupe A

MICHELI Sébastien

Rapport de projet : TSI

Introduction :

Ce projet a pour but de nous faire manipuler les fonctionnalités d’OpenGL à travers la réalisation d’un jeu. Nous avons décidé de nous inspirer du jeu Geometry Dash : le but du jeu est d’aller au bout d’un niveau en sautant pour éviter les obstacles. Votre personnage (un cube) se déplace tout seul et il faut le faire sauter pour passer au-dessus des différents obstacles sur votre route.

Voici donc un rendu du jeu à l’écran :

Une image contenant dôme, bâtiment

Description générée automatiquement

Le cube de gauche est votre personnage.

Le petit rond noir est un tremplin, passer dessus vous permettra de faire un saut plus haut que celui de base de votre personnage.

Le triangle rouge est un pic. Si vous le touchez vous avez perdu et vous retourner au début du niveau.

Le cube de droite est une plateforme sur laquelle vous pouvez sauter. Attention, si vous arrivez de face en rentrant dans la plateforme, vous mourrez. Il faut donc bien faire attention à arriver au-dessus de celle-ci.

Pour lancer le jeu, il suffit d’exécuter le programme *main*. Si vous êtes sous Linux, entré « y » dans le terminal pour ouvrir la fenêtre du jeu, sinon sous Windows entré « n ».

Commandes :

Touches dans l’écran titre :

- s : lance la sélection du niveau

- echap : ferme le jeu

Touches dans la sélection du niveau :

- flèche > : sélectionne le niveau suivant

- flèche < : sélectionne le niveau précédent

- entrée : lance le niveau

- echap : ferme le jeu

Touches durant la partie :

- barre espace : fait sauter le cube

- echap : ferme le jeu

Structure du code :

Le code est composé de différents programme permettant chacun d’effectuer une partie précise du jeu. Nous allons donc vous expliquer les parties importantes de chacun d’entre eux.

Programme *main* :

Il permet de lancer le jeu en appelant tous les autres programmes et en définissant les paramètres dont ils ont besoin. Il permet également de l’affichage du texte pour l’écran titre ainsi que pour la sélection du niveau.

Programme *background* :

On définit le fond d’écran de la zone jouable. Pour cela on définit la taille qu’elle doit occuper ainsi que l’image à mettre sur cette zone.

Une image contenant dôme, bâtiment

Description générée automatiquement

Cette zone se décompose en trois parties :

- Les parties 1 et 3 qui sont faces à la caméra, faisant office de mur.

- La partie 2 qui relie ces deux murs et sert de sol pour notre jeu. C’est sur cette partie que le joueur et les obstacles seront situés

Programme *cpe3d* :

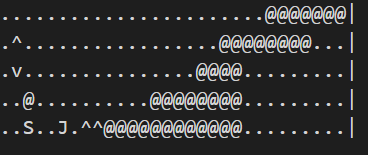
Programme *glutils* :

Ce programme comporte des fonctions utilisées dans d’autres programme. Tout d’abord il permet la compilation du vertex et fragment shaders.

Programme *level* :

Ce programme comporte 2 classes : une pour gérer une potentielle erreur lors de l’exécution, et l’autre pour définir les fonctions utiles à l’exécution du jeu.

La première étape est de chargé le niveau que le joueur a choisi. Chaque niveau correspond à un fichier texte dont les symboles écrit à l’intérieur vont être associé à un obstacle. Voici à quoi ressemble une partie du fichier texte du niveau de test :



Chaque ligne représente la position (selon y) où sera placer un objet. Ils sont ensuite affichés dans l’ordre des symboles. Chaque symbole va correspondre à quelque chose à afficher :

- . : on ne place pas d’obstacle.

- J : on place un tremplin (J pour Jump).

- @ : on place une plateforme.

- <, v ou ^ : on place un pic orienté selon le sens du symbole.

- | : fin du niveau.

- S : position de départ du personnage.

Durant la partie tous les obstacles passant derrière le joueur, et donc ne pouvant plus être croisés, vont être supprimés.

Programme *mesh* :

Programme *obstacles* :

On définit une classe pour chaque type d’obstacles. La première classe correspond aux pics qui vont être les principaux obstacles. La fonction contient toutes les données nécessaires pour les placés durant le jeu. Elles sont toutes identiques hormis sur leur orientation qui pourra varier.

Programme *player* :

Cette partie est dédiée aux fonctionnalités rattachées au joueur. La première fonction sert à placer le joueur au début du niveau (position S dans le fichier texte du niveau).

On définit les caractéristiques du jeu : la vitesse de déplacement du joueur (en réalité celle des obstacles car le joueur est immobile), la réapparition au début du niveau en cas de mort, la hauteur du saut du joueur lors d’un saut normal (barre espace) et lors du passage sur un tremplin. Pour cette dernière partie il faut être capable de détecter les collisions.

La fin du programme a donc pour but de détecter les collisions. Pour cela on teste si des hitbox se superposent. Voici un exemple avec le schéma suivant :

Une image contenant texte

Description générée automatiquement

Personnage (carré bleu clair), pic (triangle), plateforme (carré bleu foncé), hitbox du pic/plateforme (carré noir).

Situation de gauche : Nous avons défini la hitbox des triangles avec des carrés car c’est plus simple de les définir de cette manière. La hitbox du carré est exactement ce carré. Ainsi s’il rentre en contact avec la hitbox du pic il mourra.

Situation de droite : La hitbox de la plateforme est légèrement plus basse pour permettre au joueur de sauter sur celle-ci. Ainsi s’il saute sur la plateforme il ne sera pas en contact avec la hitbox est ne mourra pas. Cependant s’il arrive de face il y aura contact et il sera bien mort.

Programme *reflist* :

Programme *viewerGL* :

Conclusion :

Récap de ce qu’on a fait.