Rapport - Projet Moteur de jeux

Brian Delvigne & Loïc Kerbaul
Master 1 IMAGINE
Université de Montpellier

Mars - Mai 2024

1 Dépôt GitHub et compilation

Notre code est disponible sur ce dépôt GitHub (cliquez ici). Sur celui-ci, vous y trouverez 4 branches, mais seule la branche *main* est à prendre en compte, les autres n'ont servi que pour des tests. Voici la manière de procéder afin d'exécuter notre code :

- Une fois le dépôt GitHub cloné, placez-vous dans le répertoire principale, et créez le répertoire *build*.
- A l'intérieur de ce nouveau répertoire, lancez la commande *cmake ..*, qui produira un Makefile.
- Toujours dans le répertoire *build*, compilez le projet avec la commande *make*, ce qui produira l'exécutable *main*.
- Exécutez enfin la commande *./main* afin de démarrer l'exécutable.

2 Panneau de contrôle ImGui

A l'ouverture de notre application, vous trouverez un panneau de contrôle ImGui. Celui-ci peut être afficher/retirer en appuyant simplement sur la touche 'f' du clavier. Vous trouverez sur ce panneau de contrôle les informations suivantes (voir le résultat sur la figure plus bas) :

- Style de la fenêtre ImGui (parmis les 3 disponibles)
- Nombre de FPS
- Position du joueur sur le terrain
- Vitesse de la caméra (n'impacte que la caméra orbitale, libre, et celle contrôlée par la souris).
- Vitesse du joueur
- FoV de la caméra
- Afficher en rendu filaire
- Afficher/Retirer l'HUD du joueur
- Possibilité de choisir entre le mode créatif et le mode survie. En mode créatif (contrairement à la survie) vous pourrez casser les blocs instantanément, le joueur ne subira aucun dégât, et aura une endurance infini.
- Passage dans le mode éditeur (permettant de créer des nouvelles structures)
- Contrôle de la taille du terrain généré (en longueur et en largeur uniquement)

- La seed utilisée pour la génération du terrain
- Le nombre d'octave utilisé pour la génération (plus ce nombre est haut, plus le relief du terrain sera important)

Dans cette même fenêtre ImGui, vous avez aussi la possibilité de choisir entre différents types de caméras :

- **Caméra orbitale** : En rotation autour d'un point fixe, situé au centre du terrain
- **Caméra libre** : Déplacement au clavier (ZQSD) vers la droite/gauche et avant/arrière, sans possibilité de modifier l'orientation de la caméra
- **Caméra souris** : Changement de l'orientation grâce à la souris, et déplacement au clavier (ZQSD) selon l'orientation donnée à la caméra
- **Caméra Player** : Changement de l'orientation grâce à la souris, et déplacement au clavier (ZQSD). Ce mode de déplacement est soumis à la gravité et aux collisions avec les blocs.

Figure 1: **Fenêtre ImGui de notre projet**

La touche 'e' permet d'ailleurs, lorsqu'on est en mode de jeu normal (c'est-à-dire hors mode éditeur de structure), de passer de la caméra Player à la caméra libre, et inversement. Attention cependant, lorsque la caméra souris est utilisée, le joueur est toujours soumis à la gravité, et subit les dégâts causés par d'éventuelles entités.

3 Liste des contrôles

- A la souris :
  - **Clique gauche :** Casser un bloc ou frapper une entité. Lorsque le clic gauche est maintenu, le joueur peut viser un nouveau bloc pour le détruire, sans avoir besoin de lacher le clic. Attention : on distingue 2 modes de jeu pour le joueur, qui sont le mode "survie" et le mode "créatif". Dans le premier, casser un bloc prendra un certain temps durant lequel une animation se jouera sur le bloc que l'on est en train de casser, tandis qu'en mode "créatif", casser un bloc est instantané.
  - **Clique droit :** Poser un bloc. Contrairement au clic gauche, il est nécessaire de relâcher le clic droit après avoir posé un bloc, si on souhaite en poser un autre.
  - **Molette :** Permet de changer le bloc que le joueur a en main (celui qu'il posera). Sur la barre de bloc de l'HUD (que l'on appelle hotbar), un sélecteur se déplace en fonction de la molette afin d'indiquer le bloc actuellement en main.
- Au clavier (lorsque l'on contrôle le joueur et qu'on est donc avec la caméra Player) :
  - Les touches 'z' et 's' permettent de se déplacer d'avant en arrière, et les touches 'q' et 'd' de gauche à droite.
  - Maintenir la touche 'Shift' pour courir
  - Appuyer sur 'Espace' pour faire sauter le joueur
  - Touche 'h' : Faire apparaître/disparaître le panneau de contrôle ImGui
  - Touche 'e' : Permet de passer de la caméra Player (celle où on contrôle le joueur) à la caméra libre, et inversement (voir quelle checkbox est cochée dans la fenêtre ImGui pour savoir quelle caméra est activée)
  - Touche 'p' : Active/désactive l'affichage des effets d'ombres sur les blocs
  - Touche '+' et '-' : Elles permettent de naviguer dans la hotbar afin d'avoir accès à d'autres blocs. Ainsi, tous les blocs possibles (soit un total de 37 blocs différents) sont accessibles depuis la hotbar. Si la machine sur laquelle le code est exécuté ne dispose pas de pavé numérique, on peut aussi utiliser 'n' et 'b' (respectivement à la place de '+' et '-').

4 Quelques détails sur les fonctionnalités implémentées à notre moteur

- Concernant l'HUD :
  - La barre rouge représente la vie du joueur, et la verte représente son endurance, qui lui permet de courir en maintenant 'Shift'.
  - Comme expliqué plus haut, sur la hotbar (la barre des blocs en bas de l'écran), un sélecteur indique le bloc actuellement dans la main du joueur, c'est-à-dire celui qu'il posera au prochain clic droit. Pour rappel, les touches '+' et '-' permettent d'avoir accès à d'autres blocs que ceux présent dans la hotbar initialement.
- A propos de la génération procédurale :
  - Le terrain est divisé en chunk, chaque chunk étant un espace de $32 \times 32 \times 32$ blocs.
  - Lors de la génération du terrain, plusieurs types de biomes peuvent être utilisés (le type de chaque chunk est défini aléatoirement), sachant que chaque biome utilisent des blocs différents pour le terrain, ainsi que des structures différentes.
  - L'entièreté du terrain est déterminée à partir d'une graine de génération, que l'on appelle seed. Ainsi, par exemple sur 2 machines différentes, avec une même valeur de seed, le terrain qui sera généré sur chacunes de ces machines sera identique (les reliefs du terrain, les biomes utilisés, les structures, et la génération des minerais seront identiques).
  - La génération du terrain se déroule en plusieurs étapes :
    * On commence par déterminer la hauteur des blocs à la surface du terrain en utilisant un bruit de Perlin (les types des blocs utilisés pour la surface diffèrent selon le biome utilisé). En chaque point du terrain, on place des blocs sous le bloc d'hauteur maximale, de manière à remplir le chunk sous la surface du terrain.
    * Une des optimisations que nous avons utilisé est de définir si un bloc est visible ou non par le joueur au moment où il est généré. Ainsi, avec l'étape précédente, des trous peuvent apparaître lorsque la différence de hauteur entre 2 blocs adjacents est supérieur à 1. En réalité, ce ne sont pas vraiment des trous puisque des blocs sont bien présent, mais invisible. La seconde étape consiste donc à redéfinir la visibilité de ces blocs lorsque cela est nécessaire.
    * On passe ensuite à la génération de minérais dans les couches inférieures du terrain. On modifie ensuite à nouveau le terrain généré afin de faire apparaître des filons de minerais là où des blocs de minerais ont été généré. Au maximum, un filon pourra occuper un espace de $3 \times 3 \times 3$ blocs.
    * Pour finir, la dernière étape consiste à placer des structures (en fonction du biome utilisé) à la surface du terain. De la manière dont nous l'avons fait, les structures apparaissent aléatoirement et peuvent se

superposer. Ceci est volontaire de notre part, et aurait pu être assez facilement empêché. Par exemple, il aurait été possible d'empêcher l'apparition de nouvelles structures là une structure aurait déjà été générée, en utilisant une grille de cellule à 2 états (occupée ou non).

  * A la dernière couche du terrain, des blocs de bedrock sont générés, qui sont indestructible (en mode "survie" et "créatif", le joueur ne peut pas les casser).

- L'éditeur de structure :
  - Nous avons ajouté la possibilité de pouvoir créer des structures utilisables lors de la génération du terrain.
  - Une checkbox dans le fenêtre ImGui permet de rentrer dans ce mode éditeur. Celui-ci crée un espace d'un unique chunk (donc $32 \times 32 \times 32$ blocs) avec un bloc de pierre au centre de ce chunk, qui sert de référence pour construire une nouvelle structure.
  - Les contrôles dans le mode éditeur sont les mêmes que ceux dans le mode de jeu, la seule différence est que la caméra Player ne sera plus disponible.
  - Une fois la structure construite dans l'éditeur, un champ est disponible dans la fenêtre ImGui, dans lequel il faut renseigner un nom qui sera utilisé pour sauvegarder la structure dans un fichier. Ce fichier sera placé dans le répertoire *Structures* (depuis la racine du répertoire principal). Une fois le nom saisi, il faut appuyer sur le bouton en dessous du champ de texte pour créer le fichier de la structure.
  - Lorsqu'un fichier pour une structure sera crée, si un fichier du même nom existe, celui-ci sera écrasé par la nouvelle structure. Dans notre moteur, nous n'avons fait aucune vérification sur le nom saisi dans la fenêtre ImGui, il n'est donc pas impossible d'avoir certaines erreurs qui stopperait l'exécution du programme dans certains cas (avec l'utilisation de caractères spéciaux par exemple).
  - Une fois, la nouvelle structure construite et enregistrée, il n'est pas possible de revenir dans le mode de jeu. Il faut forcément stopper l'exécution puis relancer le programme.
  - Ensuite, pour l'étape de génération du terrain, il suffit de modifier certaines valeurs dans le code afin d'inclure de nouvelles structures, modifier leur taux d'apparition, etc...

- Dans le mode de jeu
  - La barre rouge représente la vie du joueur, sachant qu'il peut subir des dégâts (uniquement en mode "survie") en chutant de trop haut où en se faisant attaquer par certaines entités (dans notre cas le zombie).
  - La barre verte représente l'endurance du joueur. Tant qu'il lui reste de l'endurance, le joueur peut courir en maintenant la touche "Shift". Lorsque le joueur relâche cette touche, la barre se remplit jusqu'à ce qu'elle soit pleine.

  - Le joueur a la possibilité de casser des blocs, ainsi que d'en poser de nouveau (uniquement à une distance fixe) en fonction du type de bloc qu'il a en main (défini par le sélecteur de la hotbar, controlé avec la molette de la souris).
  - Il est aussi possible d'activer/désactiver un effet d'ombre en appuyant sur la touche 'p'. La luminosité de chaque texture du bloc est alors calculée en fonction de la face sur laquelle elle est appliquée.
  - En maintenant le clic gauche, le joueur peut faire disparaître les entités qui sont suffisamment proche de lui. Nous n'avons pas fait en sorte que le joueur ait besoin de viser les entités qu'il souhaite faire disparaître (par manque de temps).

- Fonctionnement des entités (contrôlées par des agents) :
  - Dans notre moteur, nous avons ajouté la possibilité de créer des pavés rectangulaire afin de former des entités. On peut de plus ajouter des animations à ces entités.
  - Nous utilisons un graphe de scène afin de représenter chaque entité par un noeud central. Sur celui-ci les transformations sont effectuées (translation et rotation dans notre cas) afin de pouvoir déplacer et animer l'ensemble de l'entité.
  - Nous avons donc implémenté 2 types d'entités : les zombies et les cochons (sur lesquels nous avons d'ailleurs appliqué des textures). Les cochons, complétement passif, ne font que se déplacer aléatoirement. Tandis que les zombies, eux, poursuivent le joueur (en s'orientant dans sa direction) si celui-ci se trouve dans leur zone de détection (on calcule pour cela simplement la distance entre le zombie et le joueur). Si le joueur est hors de cette zone, le zombie se déplace aléatoirement (comme le cochon).
  - Les entités utilisent tout comme le joueur une hitbox (c'est d'ailleurs la même classe et les mêmes fonctions qui sont utilisées par le joueur et les entités) afin de déterminer les collisions qui s'appliquent. Elles subissent donc l'effet de la gravité et ont des collisions latérales. Nous avons fait le choix de faire en sorte que les entités ne puissent pas sauter, ainsi nous n'avons pas eu besoin de vérifier les collisions vers le haut (si nous l'avions fait, cela aurait été assez simple puisqu'il suffisait de réutiliser ce que nous avons fait pour le joueur).

- Utilisant d'une librairie audio :
  - Afin de pouvoir jouer des sons dans notre moteur, nous avons utilisé la librairie Miniaudio, une librairie minimaliste qui dispose d'une API haut niveau assez simple à manipuler.
  - Grâce à cette librairie, nous avons fait en sorte de jouer des musiques de fonds, parmis les 5 musiques que nous avons mis dans notre projet (dans le répertoire *Sound/BackgroundMusic*). A la fin de chaque musique, un callback est appelé (c'est la librairie Miniaudio qui permet une telle fonction) qui s'occupe de lancer une nouvelle musique.

– De plus, nous avons aussi ajouté des effets sonores : Lorsque l'on casse ou pose un bloc, lorsque le joueur prend des dégâts, etc... Tous ces sons sont fournis là encore dans le répertoire *Sound* du projet.