Lycée Réaumur - Laval (53)
Lycée de l'Hyrôme - Chemillé-en-Anjou (49)

24h du code 2024 : MineGen

Introduction

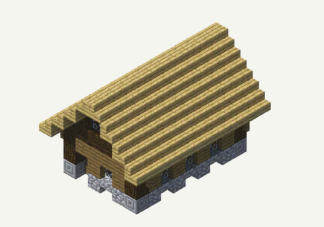
Vous connaissez quasiment tous le jeu Minecraft mais jouons les innocents et allons nous renseigner sur le sujet. Selon notre ami Wikipédia :

> "Minecraft est un jeu vidéo de type aventure « bac à sable » (construction complètement libre) développé par le Suédois Markus Persson, alias Notch, puis par la société Mojang Studios. Il s'agit d'un univers composé de voxels (blocs) et généré de façon procédurale, qui intègre un système d'artisanat axé sur l'exploitation puis la transformation de ressources naturelles (minéralogiques, fossiles, animales et végétales).
>
> [...]
>
> Minecraft plonge le joueur dans un monde créé de manière procédurale, composé de voxels (des cubes) représentant différents matériaux comme de la terre, du sable, de la pierre, de l'eau, de la lave ou des minerais (du fer, de l'or, du charbon, etc.) formant diverses structures (arbres, cavernes, montagnes, temples). Le monde est également habité par des animaux ou monstres tels que des vaches, des moutons, des zombies et des squelettes. Le joueur peut modifier cet ensemble à volonté en y ajoutant ou supprimant des blocs et en tentant de survivre le plus longtemps possible, ces possibilités lui permettant ainsi de bâtir des constructions avec une grande liberté, rappelant les jeux de création Lego.
>
> [...]
>
> Un des grands intérêts du jeu réside dans la création procédurale du monde : chaque carte est générée aléatoirement au fur et à mesure que le joueur avance, selon une graine de monde générée suivant l'heure de l'ordinateur. Cependant, le monde est majoritairement cohérent, composé de plusieurs paysages nommés biomes (forêt, plaine, désert, jungle, toundra, taïga, marais, savane, etc.)
>
> [...]
>
> Le monde principal est composé d'une surface, dépendante de chaque biome, et de souterrains, plus ou moins remplis de pierres, où se génèrent aléatoirement des minerais, des grottes ou des cavernes. Des structures comme les donjons, temples aquatiques ou encore ancienne cité apparaissent également aléatoirement sur la carte, à la surface ou dans les souterrains."

Dans ce sujet composé de deux parties entièrement indépendantes, nous allons nous intéresser d'une part à la génération de structures sur une map Minecraft et d'autre part à la mise en place d'une plateforme web permettant l'administration de notre serveur Minecraft à distance. Aussi, nous vous conseillons de lire le sujet entièrement et de vous répartir les tâches avant de partir dans le développement de vos outils.

1ère partie : génération de structures aléatoires

Ce 1er sujet consiste à automatiser la génération d'une ou plusieurs structures (un bâtiment, une cabane, quelque chose d'artificiel) sur une map Minecraft.



Minecraft est un jeu avec une architecture client-serveur, c'est à dire que les joueurs (clients) vont devoir se connecter à un serveur dont le rôle principal est de servir les requêtes des clients. C'est ce serveur qui héberge la map et qui connait donc la position de chaque bloc et de chaque joueur à un instant t. Ce que voit chaque joueur sur son écran n'est qu'une réplication basée sur les informations transmises par le serveur.

1ère étape : mise en place du serveur Minecraft

Dans notre cas, chaque groupe devra mettre en place un serveur Minecraft pour pouvoir tester ce que vous développez en local. Dans un soucis de compatibilité avec tout le monde, le serveur ainsi que le client devront être dans leur version 1.20.4.

2ème étape : poser un bloc à partir d'une commande de jeu

Sur un serveur Minecraft, au delà du fait de jouer normalement avec le client que vous utilisez tous développé par Mojang, il est possible d'exécuter des commandes permettant d'interagir avec le jeu. Il existe une documentation complète qui répertorie toutes les commandes possibles, nous vous laissons vous renseigner.

Nous rappelons qu'une map Minecraft est constituée de blocs. Donc pour s'intéresser à la génération d'une structure, nous devons d'abord savoir comment poser un bloc automatiquement, sans le faire à la main dans le jeu. Pour cela (comme pour des tas d'autres fonctionnalités) il y a une commande qui s'occupe de poser un certain bloc (pierre, bois, terre...) à des coordonnées bien précises (x, y, z).

3ème étape : automatisation avec le protocole RCON

Une fois que vous savez poser un bloc grâce à une commande dans Minecraft, il serait préférable pour automatiser la pose des blocs d'externaliser l'exécution de cette commande depuis un script Python (par exemple). Pour cela, vous allez utiliser le protocole RCON (Remote CONsole) qui est basé sur TCP et qui permet de créer une socket (un tuyau d'échange de messages entre un client et un serveur). Il existe dans de nombreux langages de programmation des bibliothèques RCON permettant d'envoyer un message à un serveur. C'est pour cette raison que nous ne vous imposons pas de langage particulier. Nous vous conseillons de le faire en Python mais vous pouvez très bien utiliser un autre langage que vous maitrisez si vous en avez envie.

De la même manière que pour poser un bloc, nous vous laissons vous documenter sur RCON. Le but de cette tâche est de pouvoir exécuter n'importe quelle commande depuis un script Python externe au serveur.

4ème étape : génération automatique des structures

A partir de maintenant, vous savez poser un certain bloc à des coordonnées x y z sur la map grâce à un script Python. Nous vous conseillons de bien structurer votre code à l'aide de fonctions ou de classes.

Nous rappelons que le but du sujet est de pouvoir générer des bâtiments aléatoirement mais en respectant à chaque fois un cahier des charges bien précis. Pour vous donner un exemple sur Minecraft, un mineshaft (donc une mine) respecte à chaque fois une structure bien précise, de sorte à ce qu'on la reconnaisse lorsque nous en rencontrons une. En revanche, elle est à chaque fois générée aléatoirement et donc différente (ce n'est jamais la même). Nous allons essayer d'en faire de même de manière plus modeste avec des bâtiments à la surface.

Votre algorithme devra être capable de prendre des informations en entrée, comme la superficie du bâtiment, le nombre d'étages ou encore le type du toit à générer. Pour cela, vous allez devoir lire un fichier sous format JSON, qui est un type de format permettant de structurer des informations. Nous vous laissons vous documenter sur comment lire et décoder un fichier JSON dans le langage de votre choix.

Ce fichier JSON sera plus ou moins enrichi en fonction de votre avancement dans le projet. Plusieurs échelons doivent être franchis pour arriver à un résultat final.

Échelon 1

Réaliser une maison avec comme contraintes :

- le nombre de blocs en largeur tiré au sort entre deux valeurs
- le nombre de blocs en longueur tiré au sort entre deux valeurs
- une porte (sur le pignon)
- deux fenêtres par coté (sur la longueur)

Voici la structure du fichier JSON de l'échelon 1 que vous devrez lire (avec des valeurs d'exemple) :

```
{
  "coords" : [103, 62, 205] # coordonnées de la structure à générer
  "range_x": [10, 20], # nombre de blocs en largeur tiré au sort entre ces deux valeurs
  "range_z": [10, 20], # nombre de blocs en longueur tiré au sort entre ces deux valeurs
  "floor_size": 4 # nombre de blocs par étage
}
```

Échelon 2

En plus des contraintes de l'échelon 1, rajouter la possibilité de choisir le nombre d'étage à générer sur la maison. Chaque étage devra être relié par un escalier avec son étage n-1. Son fichier JSON sera le suivant :

```
    {
      "coords" : [103, 62, 205] # coordonnées de la structure à générer
      "range_x": [10, 20], # nombre de blocs en largeur tiré au sort entre ces deux valeurs
      "range_z": [10, 20], # nombre de blocs en longueur tiré au sort entre ces deux valeurs
      "floor_size": 4, # nombre de blocs par étage
      "floors": 2 # nombre d'étages
    }
```

Échelon 3

En plus des contraintes de l'échelon 2, rajouter la possibilité de choisir :

- le nombre de fenêtre à rajouter au global sur la maison
- si on souhaite un toit plat ou pentu

Le fichier JSON correspondant :

```
    {
      "coords" : [103, 62, 205] # coordonnées de la structure à générer
      "range_x": [10, 20], # nombre de blocs en largeur tiré au sort entre ces deux valeurs
      "range_z": [10, 20], # nombre de blocs en longueur tiré au sort entre ces deux valeurs
      "floor_size": 4, # nombre de blocs par étage
      "floors": 2, # nombre d'étages
      "flat_roof": True, # si on souhaite un toit plat ou pendu
      "windows": 4 # nombre de fenêtres au total
    }
```

Échelon 4

En plus des contraintes de l'échelon 3, rajouter la possibilité de créer un hameau de n maisons, n étant compris entre deux valeurs. Les maisons doivent également êtres placées de manière aléatoire. Le fichier JSON correspondant :

```
    {
      "coords" : [103, 62, 205] # coordonnées de la structure à générer
      "range_x": [10, 20], # nombre de blocs en largeur tiré au sort entre ces deux valeurs
      "range_z": [10, 20], # nombre de blocs en longueur tiré au sort entre ces deux valeurs
      "floor_size": 4, # nombre de blocs par étage
      "floors": 2, # nombre d'étages
      "flat_roof": True, # si on souhaite un toit plat ou pendu
      "windows": 4, # nombre de fenêtres au total
      "n": [2, 5]
    }
```

Note : vous pouvez bien sûr (et c'est d'ailleurs fortement encouragé) rajouter des paramètres, mais vous devrez en informer votre jury lors de l'évaluation, pour qu'ils puissent construire un fichier JSON adapté.

2ème partie : administration du serveur Minecraft

Pour ce 2ème sujet, nous allons vous demander de créer une plateforme web permettant d'administrer votre serveur Minecraft. Pour cela, vous allez utiliser le même protocole RCON que dans la 1ère partie, qui va vous permettre d'effectuer des actions à distance non pas depuis un script Python mais depuis un serveur web cette fois-ci (en PHP par exemple).

De la même manière que pour la 1ère partie, vous allez devoir respecter des échelons de niveau pour avancer.

Échelon 1

Depuis votre site web, nous devons être capable de :

- mettre la nuit sur le serveur
- mettre le jour également
- envoyer un message sur le tchat

Échelon 2

En plus des contraintes de l'échelon 1, nous devons pouvoir :

- lister les joueurs connectés
- bannir/dé-bannir un joueur
- le déconnecter
- le tuer
- lui chuchoter quelque chose
- changer son mode de jeu (créatif / survie)

Échelon 3

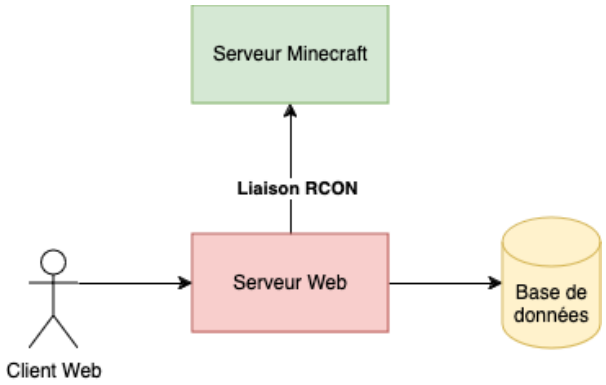
En plus des contraintes de l'échelon 2, nous devons pouvoir donner un item à un joueur (proposer une liste des items disponibles par exemple).

Échelon 4

En plus des contraintes de l'échelon 3, vous devez intégrer sur votre plateforme d'administration un système d'authentification par email / mot de passe afin de limiter l'accès aux personnes autorisées.

Ce n'est pas une liste exhaustive, vous pouvez (et devez dans la mesure du possible et de vos idées) rajouter des fonctionnalités. Toute prise d'initiative sera valorisée dans le classement final. L'esthétique du site aura également une grande part dans la notation.

Ci-dessous, vous trouverez un schéma très simple de la mise en place de vos différents éléments.



Le challenge

Le challenge final est assez simple. Vous allez être répartis en poules d'équipes : une équipe qui joue son algorithme, et les autres qui évaluent. Les équipes évaluatrices devront se mettre d'accord sur un fichier JSON commun à donner à l'équipe évaluée. Cette dernière devra, sous les yeux des deux autres équipes exécuter son programme. Les équipes évaluatrices donnent chacune une note, en fonction du respect de leur cahier des charges.

Pour des raisons de logistique, vous-vous connecterez pour cette épreuve au serveur Minecraft des 24h du code accessible via l'adresse IP suivante : **82.66.47.215** (également joignable sur **24hducode.freeboxos.fr**) sur le port RCON par défaut (**25575**) et avec le mot de passe suivant : **choucroute**.