Terraria

1 Le jeu

Le jeu se présente sous la forme d'un terraria. C'est à dire un monde 2D composé de cubes minables. Le joueur possède un inventaire, avec de base une pioche et une épée. La pioche permet de récolter des blocs et l'épée permet de fraper les ennemis. Le joueur à 5 pv et perd de la vie quand il se fait attaquer. Le joueur peut se protéger avec son bouclier. Le joueur peut placer les blocs qu'il a ramassé

1.1 Controles

```
A            : items précedent
E            : items suivant
Q            : Aller à gauche
D            : Aller à droite
espace       : sauter
click droit  : utiliser l'objet
click gauche : se protéger
```

2 structure du code

2.1 structure générale

Ce projet est structuré selon un modèle vue contolleur un peut modfié. En effet il y a 5 fichiers principaux. Tout d'abord le Controller.js qui va s'occuper de vérifier si une touche est préssée ou non. Ensuite, l'engine qui ne s'occupe que d'affciher le jeu au bon nombre de frame par seconde. Display.js s'occupe de dessiner tout les éléments de la scene dans un buffer puis d'afficher ce buffer à l'écran. Game.js contient tout les éléments du jeu, en l'occurence, la map, l'inventaire, le joueur et les ennemis. Le fichier main.js lie ensemble ses 4 fichier afin de les faires fonctionner efficacement.

2.2 main.js

Le main est composé de différents partie. Tout d'abord un assetManager qui va load toute les textures puis va lancer le jeu. Il est aussi composé de 2 fonctions importantes. la fonction render, qui va demander à la classe Render de dessiner chaque element de la scene. Mais aussi de la fonction update, qui va tester tout les inputs possibles et si cet input est actif, il va lancer l'action correspondante. Puis il va appeler la fonction update de la classe game.js

2.3 game.js

La fonction principale de Game.js est al fonction update dans la class World. Elle va update tout les éléments du jeu. Tout d'abord elle va mettre à jour la position de tout les ennemis. Elle va ensuite faire attaquer les ennemis si ils sont assez proche du jouer. Elle va aussi déclencher l'attaque du joueur si l'input click gauche est activé et si le joueur a son épée en main. Ensuite elle va mettre à jour la position de tout les ennemis et du joueur. puis elle va appeler le collider sur ces 2 dernier afin de vérifier qu'ils ne sont pas en collision avec la map, et, si oui, leur position est mit à jour. Et pour terminer, l'animation du joueur et des ennemis est mit à jour.

2.4 collideObject

collideObject est la fonction principale de collision. Pour s'assurer que le joueur n'est pas en collision avec une plateforme. Pour ce faire, ma fonction va récupérer la position du jouer, et calculer tout les blocs dans lequel il est. Si il se trouve dans un bloc solide non vide, alors la position du joueur va etre changée pour qu'il n'y ai plus de collision.

2.5 Display

Display a pour role de dessiner chaque element du jeu sur une image puis de coller l'image sur l'écran. Pour cela la classe Display est composée de plusieurs fonctions. Tout d'abord la fonction draw map qui a été la plus compliquée à implémenter. En effet, pour que le jeu le lag quand la map est grande, il n'était pas possible de dessissiner toute les tiles de la map une par une. C'est pourquoi la fonction va dessiner uniquement les blocs qui sont affichés dans la caméra afin de sauver de la puissance de calcul. Une autre fonction importante est la fonction draw object qui va dessiner soit le joueur, soit un ennemi à une position voulue. La fonction draw interface dessine une barre d'inventaire vide, puis chaque item de l'inventaire et enfin place un curseur à l'endroit ou se trouve l'item actif.

2.6 generate map

Pour la création de la map, une grotte type a été crée et elle est collée aléatoirement sur la map. De plus, à chaque grotte, un ennemi est ajouté dedans