Rapport PC2R

Auteur:
Cyann Donnot

April 15, 2019

1 Mode d'emploie

1.1 Serveur

Il suffit de lancer la commande :

python Server.py

pour modifier le port ou l'hote, il faut aller dans le fichier python et modifier dans le code la dernière ligne qui appelle et lance le serveur.

1.2 Client

Pour le java il s'agit d'un projet éclipse, honnêtement je ne sais pas comment le lancer hors d'éclipse, je doute que la commande suivante fonctionne.

javac src/game/Display.java

java src/game/Display

Sinon lancez-le dans eclipse.

1.3 Tickrates

Pour les variables de tickrates, elles sont présentes dans le fichier Display.java côté serveur et Server.py côté serveur. Vous pouvez modifier autres variables comme vous le souhaitez tant que vous restez cohérent entre le serveur et le client.

1.4 Commandes en jeu

Commande	Effet
Z	accélérer
Q	tourner à gauche
D	tourner à droite
ESCAP	quiter la partie
SPACE	Champ de force

Note : la commande ESCAP permet de quiter le jeu, cependant, les autres joueurs peuvent continuer à jouer sans vous.

2 Client

Pour le client, j'utilise Java Fx, les threads du client sont donc encapsulés dans une task ou sont directement dans des *"tasks"*. je ferai le racourci en disant thread à chaque fois.

2.1 Communication

Coté Client, la réception des protocoles du serveur se fait via un thread indépendant car la réception est bloquante. Cependant, l'envoie se fait un peu n'importe ou,

dès qu'une commande doit être envoyée. les commandes sont envoyées principalement par la boucle de jeu.

2.2 Lobby et connexion

Le lobby est la connexion au serveur sont gérer par le thread principal. le lobby s'arrête à la réception du protocol SESSION du serveur.

2.3 Jeu

Il y a d'abord deux gestionnaires d'événement clavier *"pressed"* et *"released"* qui permettent de savoir si les boutons sont enffoncés ou non. S'il sont enfoncés pendant une boucle de jeu, la commande associée est rélalisée. Ces trois fonctions sont des threads indépendants du main qui tournent en arrière-fond.

3 Serveur

3.1 Communication

Pour la communication, chaque client possède un thread qui lui est propre (car cette action est bloquante) qui est répertorié dans un dictionnaire python agrandi a quelques fonctions comme le remove qui gère le protocole LEFT-PLAYER ou le send_to_all qui envoie une même commande à tous les clients. Chaque client possède alors un unique lock qui est utilisé lors de la réception d'un protocole NEWCOM ou lors de la copie de cette dernière pour l'appliquer (la copie permet de libérer le lock plus rapidement)

3.2 Lobby et connexion

La librairie socket permet de donner une date d'expiration aux commandes de communication ce qui permet de sortir d'un accept après un temps donné. Ceci génère une erreur timeout qui peut être traiter sans pour autant sortir du thread en cours. Le lobby est donc traité au début du thread principal sans utiliser de thread pour faire un chrono. Les joueurs peuvent se déconnecter et se reconnecter dans le lobby, et ce dernier ne lance pas le jeu tant qu'il n'y a pas au moins un joueur et que sa durée est écoulé depuis que le premier joueur est rentré (le timer est réinitialisé si tous les joueurs quittent le lobby)

3.3 Jeu

La boucle de jeu est réalisé dans un thread qui tourne en arrière-plan et éxecute lui-même tous les calculs, elle se brise uniquement si le nombre de joueurs atteint 0.

4 Extensions

4.1 Champ de force

Le chaque joueur peut activer son champ de forces a souhait, ceci augmente votre rayon de collision sans pour autant vous y rendre plus sensible. Ceci vous permet alors de continuer dans la même direction sans partir en arrière à chaque collision.

Il n'y à pas de limitation sur l'utilisation, à vous d'être fairplay ;)