Année 2023-2024
L2 INFORMATIQUE UPCITÉ

Rapport de projet - AD2

Agar.io Groupe AD2-C

Projet réalisé par :
KACI Amel 22200653
LIU Shiqi 22211216
MA Qian En 22202681
PERRIER-BABIN Ludivine 22200483
PITEL Jérémy 22212919

Sommaire :

I / Introduction

1. Principe du jeu

Le jeu *agario* est basé sur celui du même nom qui se joue en ligne sur le site "*agar.io*". Le jeu original est un jeu en ligne multijoueur dans lequel chaque joueur contrôle une cellule et avance sur une map. Le but du jeu est de devenir la plus grande cellule en absorbant d'autres cellules et en évitant d'être absorbé par les autres cellules plus grandes. C'est un jeu qui se termine soit car le joueur décide de quitter la partie soit lorsque sa cellule est mangée et qu'il perd.

Les billes disposées un peu partout sur la map permettent de grossir un peu, le but étant de pouvoir manger les autres sur le terrain pour les faire perdre. À noter que lorsqu'on grossit la vitesse de la balle diminue. La cellule du joueur a deux possibilités pour gagner en vitesse, soit elle se split, soit elle lance de petits projectiles. Sont également présents sur la map des "virus" qui split la cellule du joueur si la taille de celle-ci est plus importante que la taille du virus.

2. Lancement du jeu

Afin de simplifier la manipulation du programme, en particulier pour le mode en ligne, nous avons décidé de mettre en place un Gradle, ainsi pour lancer le jeu on se place dans le dossier du jeu et on saisit la commande **"./gradlew runClient"**. Une fois le programme lancé, on arrive d'abord sur l'écran d'accueil depuis lequel on a la possibilité d'accéder à un menu paramètre "*Settings*" pour changer le volume du son. Toujours depuis l'accueil on demande au joueur d'entrer son nom dans la zone de saisie (si aucun nom n'est entré, un nom par défaut est attribué au joueur), et

continuer en appuyant sur “*Start*”, on arrive alors sur une nouvelle interface qui nous permet de customiser la balle, soit on choisit une couleur soit on importe son propre fichier PNG en cliquant sur “*Upload Skin*”. On continue en lançant le jeu en appuyant sur “*Start Game*”, s’offre alors à nous deux choix, le mode “*Online*” et le mode “*Solo*”.

II / Le mode Solo

Le mode Solo est le premier mode que nous avons fait afin de pouvoir retrouver les fonctionnalités du jeu original sans avoir besoin des connaissances en réseau.

1. La base du jeu

Il est composé d’une cellule de type Ball que nous jouons qui se déplace sur une map. La classe Ball hérite d’une classe parente Cercle. La caméra suit la balle selon ces déplacements, qui se font avec la souris captée par le code grâce à l’interface *MouseMotionListener*. Pour devenir de plus en plus gros, il y a un nombre fixe de petites billes sur la map. Un objet de type Bille hérite également de la classe Cercle. Le système de gestion des billes mangées garantit qu’il y a toujours en permanence le même nombre de billes sur la map : lorsqu’une bille est “mangée”, les coordonnées de celles-ci sont réinitialisées de manière aléatoire. La classe Cercle est un parent des objets de types Bille, Ball et grand-parent de Bot, ce qui permettait de faciliter l’implémentation d’une fonction détectant l’intersection entre ces objets et donc de pouvoir plus tard vérifier les collisions “dangereuses” qui font qu’un objet est mangé ou mange.

2. les différentes fonctionnalités

Pour avoir un vrai jeu fonctionnel en Solo nous avons déjà dû rajouter des bots pour avoir des adversaires à manger ou pour

perdre. Les bots apparaissent aléatoirement sur la map avec la même taille de départ que le joueur. Au début les bots se déplaçait de façon aléatoire sur la map mais au fur et à mesure ils se sont améliorés dans le sens où si une autre balle, le joueur ou un autre bots, qui serait considéré comme un danger pour lui s'approche il va vouloir l'éviter et aller dans le sens opposé. Un boit hérite d'une classe Ball puisque leur comportement est similaire dans la mesure où ils peuvent tous les deux manger et être mangés.

Ensuite nous avons ajouté en parallèle une fonctionnalité connu de base du jeu : le "Split", il permet au joueur de se couper en deux de façon équitable pour plusieurs raison, puisque lorsqu'on grossit on va moins vite, pouvoir split permet de redonner de la vitesse au joueur tout en gardant ses points car on dirige juste le double de la balle au lieu d'une seule. il faut un minimum de taille pour pouvoir se split et il suffit de faire un clic gauche avec sa souris, grâce à la bibliothèque *MouseListener* qui capte les clics de la souris.

Une autre fonctionnalité présente dans le jeu de base étant les virus, des espaces statiques sur la map qui servent a deux choses, si le joueur étant plus petit que la taille du virus alors il lui sert de cachette en étant en dessous, en revanche pour une balle d'un joueur qui serait plus gros que la taille du virus alors son nom prend sens et il split le joueur ce qui le rend plus vulnérable face au autres joueurs. Dans le jeu original, ils ont une forme ronde mais dans notre version nous les avons fait de forme octogonale pour pouvoir bien les repérer sur la map, lors du lancement de la partie ils sont placés de façon aléatoires. Dans notre version du jeu, les objets de type Ball dispose d'un attribut *isinfectedby* qui font qu'une cellule dispose d'une sorte d'immunité temporaire lorsqu'elle est infectée par un virus spécifique mais cette immunité est réinitialisée à partir du moment où la balle touche un autre virus qui peut l'infecter.

Nous avons ajouté ensuite un classement qui s'affiche en jeu et se met à jour en temps réel, on peut y voir notre joueur avec ses points et le top 10 de toutes les balles sur le terrain donc le

classement avec les bots selon leur points, donc leur taille. Tout est dans le fichier PlayerRank, qui est appelé dans le jeu dans le fichier GamePanel. Nous avons fait en sorte que la fonction de tri des balles sur la map ne prennent bien en compte que la plus grosse balle du joueur dans le cas où celui-ci a eu recours à un split à l'aide de la fonction *findLargestBall()*.

La dernière fonctionnalité que nous avons ajouté, les projectiles. Ils sont aussi dans le jeu original, pour les utiliser il suffit de faire un clic droit pour appeler la fonction qui l'applique. Pour l'utiliser c'est comme le split il faut une taille minimum pour le joueur pour ne pas avoir une balle trop petite, et surtout si le joueur a plusieurs balle on trouve d'abords la plus grosse et les projectiles partiront de celle-ci. Les projectiles sont des objets de type Bille et un constructeur spécial leur est réservé, leur diamètre est légèrement plus grand que celui d'une Bille. D'autre part, la gestion d'un projectile mangé diffère de celle d'une Bille ordinaire puisque ceux-ci ne réapparaissent pas une fois mangés.

3. les difficultés rencontrés lors du développement

Nous avons rencontré plusieurs difficultés au fur et à mesure de la progression dans le développement du jeu. Vers le début lors de la programmation de la map et du mouvement, il fallait que la caméra de l'écran suive la balle du joueur suivant ses déplacements dans la map évidemment plus grande que la taille de base de l'écran. On a d'abord essayé de passer par une map avec des tuiles projet que nous avons vite abandonné pour passer sur une map directement avec une image ou on se déplace dessus. On a donc dû utiliser des variables en plus pour pouvoir garder la balle le plus au centre possible de l'écran, qui se met à bien suivre la balle principale du joueur.

Ensuite lors de l'implémentation des bots nous avons dû mettre en place les collisions qui sont aussi utilisées pour les virus et les billes qui sont mangés. Il y a eu plusieurs versions d'une fonction qui vérifie les si une collision a bien eu lieu et ce qu'elle

doit renvoyer selon ce qui est touché par le joueur. On a donc finalement la fonction CheckColisionsULTIMATE qui utilise Intersect du fichier Cercle utilisé pour les bots et la balle du joueur pour définir quand est-ce que deux cercles se rencontrent. Malgré cela nous ne sommes pas parvenus à résoudre un problème qui survient lorsqu'un bot a une taille très importante et où la détection de la collision semble assez aléatoire.

Enfin la dernière fonctionnalité nous a pris pas mal de temps à implémenter, les projectiles, nous ont posé beaucoup de problèmes. Dans un premier temps nous avons voulu mettre les projectile avec la barre espace sauf que nous avons eu un bug de keyListener donc nous avons choisi de prendre le clic droit. Ensuite il a fallu qu'on prenne de la balle principale du joueur pour "lancer" des billes qui sont un peu plus grandes que les billes présentent de base sur la map pour pouvoir bien les différencier. Nous n'avons pas réussi à faire une animation partant de la balle vers l'endroit où le projectile est envoyé mais seulement le mettre en une fois.

III / Le mode Online

Nous avons ensuite réalisé le mode en ligne qui se base sur le mode solo mais avec l'ajout d'un serveur.

1. la mise en place du protocole

À l'initialisation, le client crée une instance de GameClient, se connectant au serveur via IP et port spécifiés(24935). Le joueur envoie son nom d'utilisateur comme première communication, ce qui est essentiel pour identifier chaque session de joueur de manière unique. On a bien gérer la gestion d'erreur pour l'adresse IP en vérifiant si le texte donné est bien d'un format d'IP

Le serveur accepte les connexions entrantes et crée un ClientHandler pour chaque client. Ce gestionnaire dédié lit le nom

d'utilisateur envoyé par le client et envoie un message de bienvenue en retour, confirmant la connexion établie.

Pour les échanges de messages, le serveur envoie des états actualisés du jeu, tels que les positions des joueurs, les billes, et les virus, en utilisant JSON pour la sérialisation des données. Et les clients envoient des données sur leurs actions, comme les déplacements et les divisions, au serveur en utilisant également le format JSON.

Les actions des joueurs sont capturées par des événements de souris dans OnlineGamePanel et envoyées au serveur sous forme de messages JSON. Et, le serveur traite ces actions, met à jour l'état du jeu, et diffuse les nouveaux états à tous les clients pour assurer la synchronisation.

Les clients peuvent envoyer des commandes spéciales comme EXIT pour se déconnecter, SPLIT_ACTION pour gérer la division du joueur, BILLE_EATEN pour gérer les billes mangé par le joueur, PlayerEaten pour gérer les joueurs mangé par le joueur. Le serveur traite ces requêtes et met à jour l'état du jeu en conséquence.

2. les fonctionnalités

Comme le jeu en solo, le mode en ligne comporte à peu près les mêmes fonctionnalités. La balle du joueur peut se split mais en revanche il peut reprendre sa propre balle pour revenir à sa taille de base. Les billes et virus sont toujours présents sur la map en même quantité et fonctionnent de la même façon. En revanche, comparé au mode solo, le mode en ligne n’a pas besoin de bots donc cette fonctionnalité a été enlevée, enfin nous n’avons aussi pas ajouté les projectiles à cause du temps.

3. les difficultés rencontrés

Comme pour Le mode solo, nous avons rencontré des difficultés dans la programmation du mode online mais assez

différents car c'était surtout lié au protocole et la relation entre le serveur et le client avec la réception des messages et leurs envoie, et faire fonctionner le tout ensemble. Nous avons donc dû repartir de plusieurs autres fichiers et faire des doublons qu'on aura plus tard condensé pour enlever quelques fichiers. Même si nous avons dû malheureusement garder quelques fonctions et fichiers qui peuvent paraître redondants. Quelque chose qui a donc été aussi compliqué fut l'implémentation des mêmes fonctionnalités qu'en solo à cause de ses doublons, ce qui explique en partie pourquoi le split est un peu différent et que les projectiles ne sont pas présents.