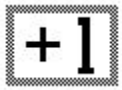
Projet de programmation - XBlast - Loïs Bilat et Nicolas Jeitziner

---

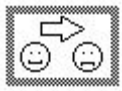
Ajouts de la version Bonus

Nous avons ajoutés à notre projet 3 Bonus supplémentaires, ainsi que de nouvelles fonctionnalités pour le serveur et le client.

Bonus ajoutés :

Bonus Life : 

Ce bonus ajoute une vie au joueur qui le prends et le rends invulnérable pendant les ticks d'invulnérabilité.

Bonus Téléportation : 

Ce bonus téléporte le joueur qui le prends sur la position d'un autre joueur aléatoirement.

Bonus Blind : 

Ce bonus fait clignoter l'écran pendant quelques secondes d'un autre joueur choisi aléatoirement.

Implémentation des Bonus:

Nous avons tout d'abord commencé par modifier la méthode applyTo(Player player) de l'énumération Bonus, de manière à permettre des bonus plus développés. Nous l'avons transformée en applyTo(Player player, Player otherPlayer). Cela sera notamment utile pour le bonus téléportation. Nous avons donc dû adapter l'appel à cette méthode dans nextPlayers de GameState. En effet, nous choisissons un joueur aléatoire parmi les autres étant encore en vie, et nous le passons à applyTo(Player player, Player otherPlayer) en tant que otherPlayer.

Bonus Life :

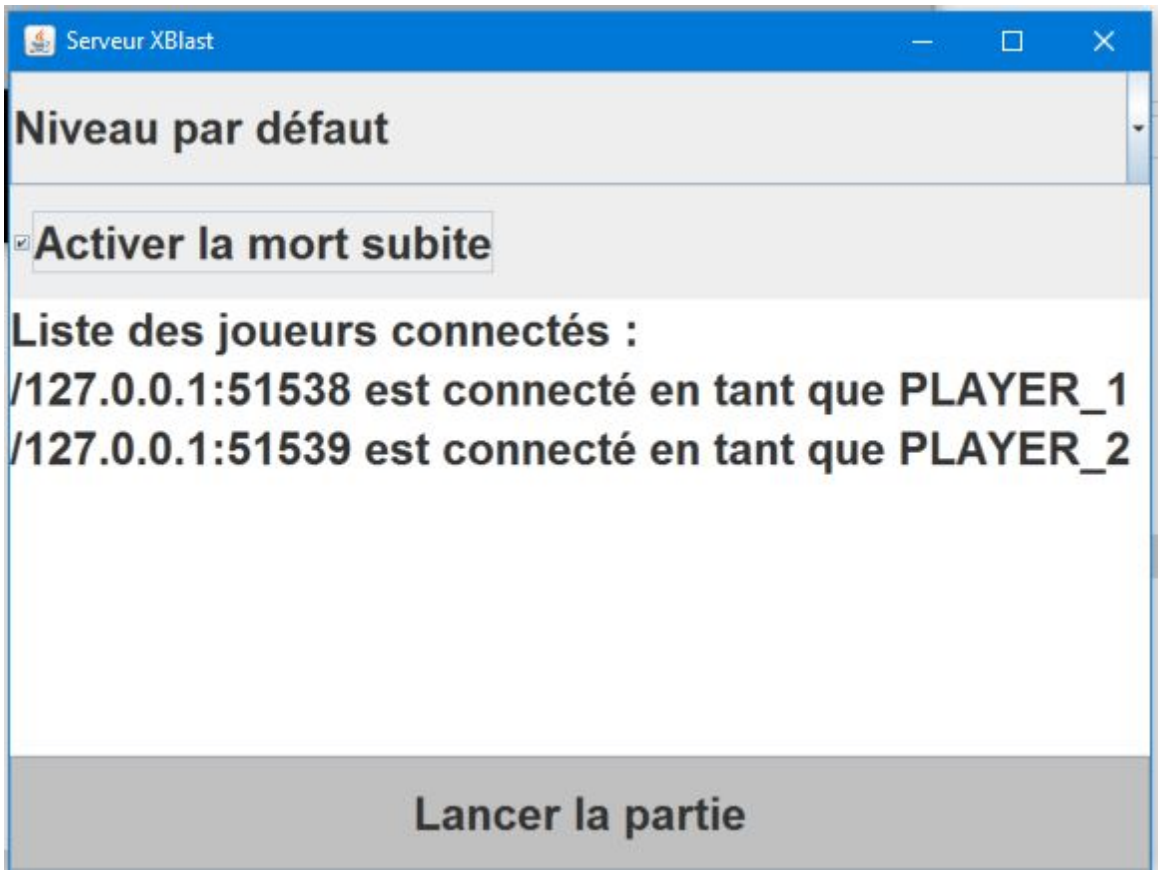
Nous avons en grande partie repris le code utilisé pour faire les bonus de base. La méthode applyTo appelle une méthode withLife(int newLife) qui retourne un joueur identique a celui passé en argument applyTo, mais en lui donnant un nombre de vie défini. De plus, on le rend invulnérable pendant un cours instant, simplement en lui recréant une Séquence de LifeStates.

Bonus Téléportation
Comme pour le bonus life, la méthode applyTo(Player player, Player otherPlayer) de ce Bonus appelle une méthode créée pour l'occasion, nommée withPosOf(Player p). Elle va créé un joueur identique à celui passé comme premier argument de appyTo(), mais en lui donnant la séquence de directedPositions du joueur passé en deuxième argument.

Bonus Blind
Pour faire ce Bonus, nous avons du commencer par modifier la classe Player du serveur en lui donnant un attribut Sq<Boolean> représentant une séquence qui donne pour chaque tick une information sur si le joueur peut voir le plateau. Nous avons par conséquent adapté les différents constructeurs, et ajouté des méthodes d'accès à la fois à cette séquence ainsi qu'au premier élément de celle-ci.
Une fois cela fait, il faut envoyer cette information au client. Nous avons donc modifié la Classe GameStateSerializer pour ajouter une byte contenant cette information juste après celui représentant l'image de chaque joueur. Le GameStateDeserializer s'occuper ensuite de traiter cette information et de la stocker dans les Player du client, eux aussi adaptés en conséquence. Au final, le XblastComponant, connaissant le joueur à qui il affiche l'interface, s'occupe de remplir l'écran avec une image noire ou blanche.

Nouvelles interfaces :

Interface du serveur : Celle-ci permet de choisir une carte parmi toutes les nouvelles cartes que nous avons crées. De plus elle permet de choisir si on veut la mort subite lors de la partie. Ceci fait apparaître des murs indestructibles toutes les 0.5 secondes sur les cellules dans l'ordre du "spiral order" en commençant par le milieu après un certain temps. L'interface affiche aussi les joueurs et leur adresse IP qui désirent se connecter au jeu. Finalement elle permet d'appuyer sur un bouton pour lancer le jeu.

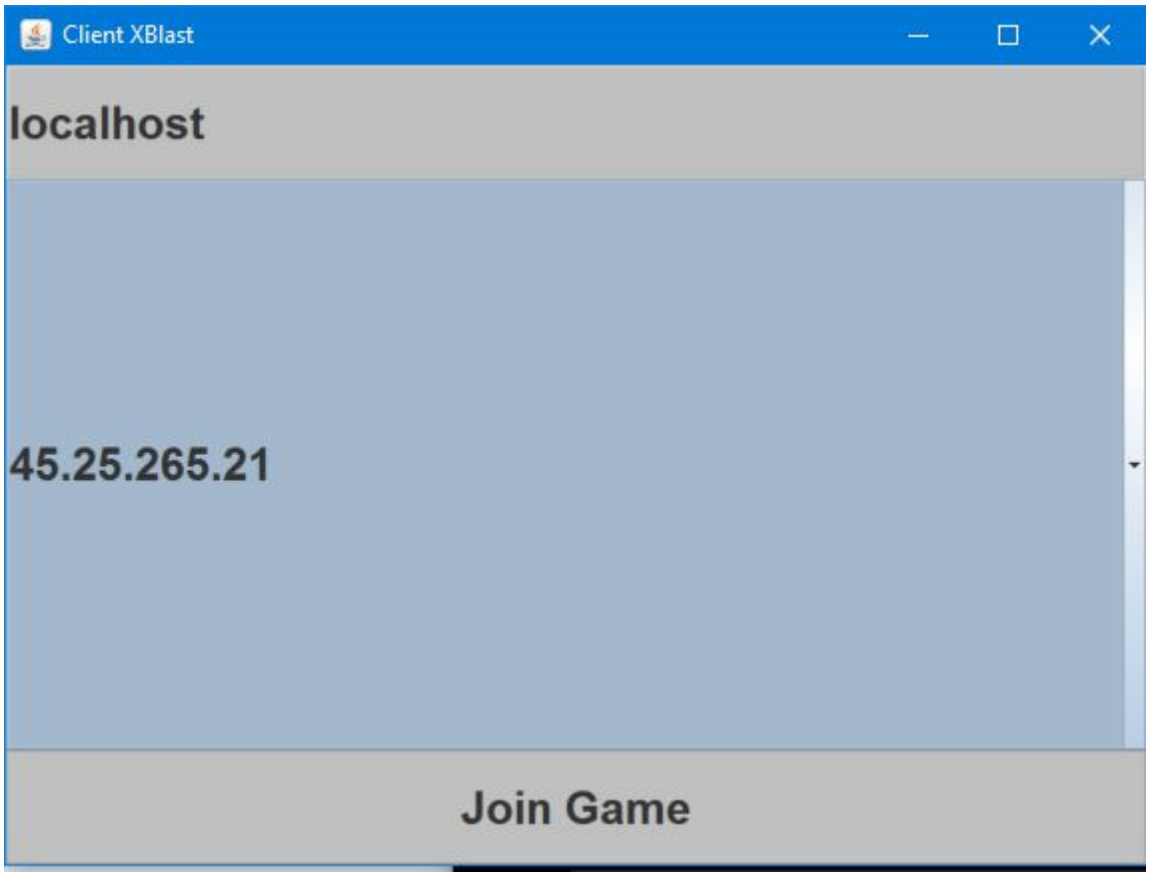


Mise en oeuvre : Nous avons créé une nouvelle méthode privée dans le Main.java du serveur qui contient le jeu en lui même. Ainsi lorsqu'on appuie sur le bouton de l'interface on appelle cette méthode et le jeu sur le serveur est lancé. L'attente de joueurs n'est pas dans cette méthode mais est toujours traitée dans la méthode main. Nous avons créé différents niveaux dans la classe Level, ceux-ci se trouvant dans une liste, et ceux-ci s'affichent dans un menu déroulant. La case permettant d'activer la mort subite change un attribut public des GameState crées à l'intérieur des différents Level.

Interface du client : Celle-ci permet à l'utilisateur de rentrer l'adresse IP du serveur et d'appuyer sur un bouton pour envoyer au serveur qu'il désire jouer. Une autre fonctionnalité que nous avons ajoutée et le fait que l'on peut choisir une adresse IP parmi les 4 dernières utilisées. L'interface prends d'abord en compte ce que l'on rentre "à la main" et seulement si l'utilisateur n'a rien rentré elle prends l'adresse choisie dans le menu déroulant.

Mise en oeuvre : Nous avons créé un fichier ip.txt dans le projet, qui stocke toutes les différentes adresses IP rentrées jusqu'à présent. Le client va récupérer les quatre dernières adresses de ce fichier et les proposer dans le menu déroulant. Si on décide d'entrer une nouvelle adresse, elle va être écrite dans ce fichier après

avoir appuyé sur le bouton, de manière à pouvoir être utilisée à nouveau la prochaine fois.



Interface de fin de partie : À la fin de la partie, c'est-à-dire ou bien si les 120 secondes se sont écoulées ou bien si il ne reste qu'un seul joueur, l'interface graphique affiche un message personnalisé selon le gagnant.

Mise en oeuvre : A chaque état du GameState, un byte est envoyé après tous les autres contenant l'information sur le gagnant de la partie. Il vaut 0 si personne n'a gagné, 1, 2, 3 ou 4 si un joueur a gagné et 5 si la partie n'est pas finie. Ce byte est enregistré dans le GameState du client sous forme d'image, grâce au deserializer. Cette image est ensuite imprimée à chaque état du jeu.