université de BORDEAUX

Rapport Projet Réseau

Ysabelle EMERY & Sébastien POUTEAU

Licence3 - Semestre5

Sommaire

Chapitre 1

Introduction

Durant notre Semestre 5 de licence, il nous a été demandé de réaliser un *Pong* en Réseau. Les difficultés de ce projet étaient de faire communiquer tous les joueurs ensembles sans utiliser de serveur central. C'est à dire que chaque joueur devait posséder son propre serveur.

Qu'est ce que le *Pong* ?

Un *Pong* est un jeu vidéo de 2 à 4 joueurs, créé en 1972, il est composé d'une raquette par joueur et d'au moins une balle. Nous attribuons un mur à chaque joueur. Un mur correspond à un côté de l'écran de jeu. Le but du jeu est que chaque joueur protège son propre mur, si la balle touche un mur, alors le joueur correspondant perd le point. Le gagnant est celui qui a obtenu le plus de points au cours de la partie.

Chapitre 2

Travail réalisé sur le *Pong*

Nous avons choisi de faire en sorte que le Pong soit le plus agréable, et le plus simple d'utilisation possible pour les joueurs et utilisateurs.

2.1 Menu

Nous avons mis en place un menu, permettant au joueur d'héberger un nouveau jeu ou bien d'en rejoindre un déjà existant. Nous avons décidé de mettre en place ce système afin de faciliter le lancement des parties et les connexions entre joueurs, permettant ainsi de rendre le jeu plus agréable à l'utilisateur.

- Hébergement : Nous demandons à l'utilisateur de rentrer le nombre de joueurs n de la partie avec $2 \leq n \leq 4$. Le jeu se lancera lorsque le nombre de joueurs connectés sera égal au nombre de joueurs demandés par le créateur de la partie. Nous avons mis en place deux modes de jeu :
  - illimité : Il n'y a pas de score maximum, et le jeu dure indéfiniment. C'est le mode par défaut.
  - limité : Le joueur doit rentrer le nombre de points que l'on doit obtenir pour gagner la partie.

  Une fois les informations validés, une nouvelle fenêtre affichera l'adresse de connexion (valable au crémi, s'il le souhaite l'utilisateur peut vérifier son adresse) et le port de connexion. Il devra transmettre ces informations aux autres utilisateurs de manière à ce qu'ils puissent rejoindre sa partie.
- Rejoindre : Nous demandons à l'utilisateur de rentrer l'adresse et le port de connexion du joueur hôte. Une page d'attente se lance en attendant que tous les joueurs aient rejoint la partie.

2.2 Connexion entre les joueurs

Nous avons décidé que tous les joueurs doivent s'être connectés avant de lancé la partie. Pourquoi ce choix ?

- Autoriser un joueur à se connecter en plein milieu d'une partie fausserait le jeu en cours. Il faudrait remettre à zéro les scores et cela pourrait être ennuyeux pour les autres joueurs.

- Cela permet d'éviter de vérifier si oui ou non il y a une connexion entrante dans la boucle principale.

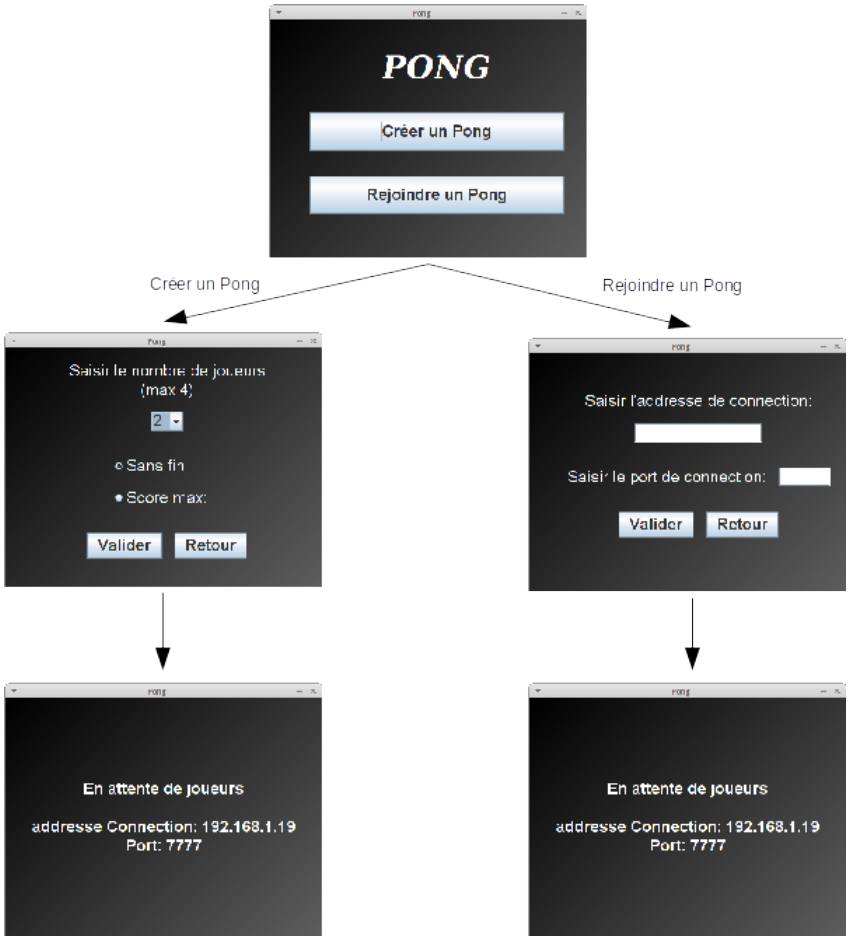

FIGURE 2.1: Interface du menu

2.3 Interface de jeu

Le positionnement et les touches directionnelles des joueurs sont différents selon le nombre de joueurs.

- 2 joueurs : les deux joueurs sont respectivement à droite et à gauche du terrain de jeu. Pour ce déplacer, ils utilisent les flèches directionnelles haut $\uparrow$ et bas $\downarrow$.
- 3/4 joueurs : les deux premiers joueurs ont le même comportement que ci-dessus, en revanche les joueurs 3 et 4 sont respectivement en haut et en bas du terrain de jeu. Pour se déplacer, ils utilisent les flèches directionnelles gauche $\leftarrow$ et droite $\rightarrow$.

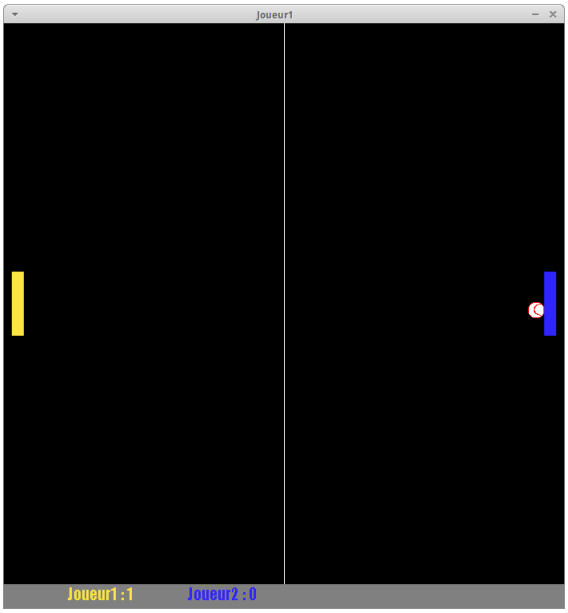

FIGURE 2.2: Interface du jeu à 2 joueurs

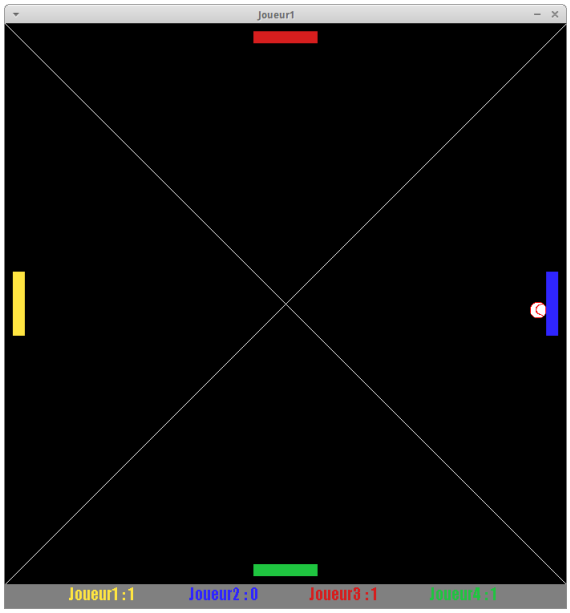

FIGURE 2.3: Interface du jeu à 4 joueurs

2.4 Procédure de fin

Nous avons mis en place une interface graphique informant les joueurs que la partie est terminée. Il y en a trois différents :

- joueur Déconnecté :

  Apparait lorsqu'il reste qu'un joueur connecté dans la partie.

FIGURE 2.4: Fin par déconnexion

- joueur gagnant / perdant :

  Ils n'apparaissent que dans une partie limité en nombre de points.

FIGURE 2.5: Fin gagnant

FIGURE 2.6: Fin perdant

2.5 Protocole de communication

Nous avons écrit tout un protocole de communication pour faciliter le dialogue entre les joueurs. De plus, ce protocole apporte toutes les fonctions nécessaires pour créer et décrypter un message. Nous faisons donc appelle à lui pour communiquer. Le joueur n'a aucune précision sur les messages transmis, donc nous pouvons modifier notre protocole sans que le joueur n'en subisse les conséquences.

2.6 Bonus et attribution

Pour pimenter le jeu, nous avons mis en place des bonus et malus. Nous avons 4 types de bonus/malus.

- Agrandissement : Agrandis la raquette du joueur.
- Réduction : Rétrécis la raquette du joueur.
- Vitesse Quick : Accélère la vitesse de la raquette du joueur.
- Vitesse Slow : Ralentis la vitesse de la raquette du joueur.

Leurs effets ont une durée limitée de 10 secondes. Un bonus/malus apparaît tous les 5 tours. Si un bonus est déjà sur le terrain ou si un bonus est encore actif alors nous ne ferons pas apparaître de nouveau bonus. Ce qui nous permet de n'avoir qu'un seul bonus à la fois. Le bonus apparaît sous la forme d'une image de

cadeau. Il rebondit sur les bords de la fenêtre afin de faciliter son interception pour tous les joueurs. Il n'est pas soumis aux collisions avec la balle.

2.7 Evité la triche

A chaque tour de boucle, un joueur enverra la position de sa raquette, de la balle et d'un bonus si il est sur le terrain. Pour éviter la triche, chaque joueur calcule les positions de tous les autres joueurs et les positions des autres éléments (balle/bonus) et va les comparer avec les données qu'il reçoit, Si les valeurs sont trop différentes alors c'est qu'un joueur triche.

Chapitre 3

Problèmes rencontrés et solutions apportés

Dans cette partie, nous expliquons tous nos problèmes et les solutions que nous avons trouvé pour résoudre ces problèmes. Pour commencer, voici un diagramme représentant l'exécution de notre programme.

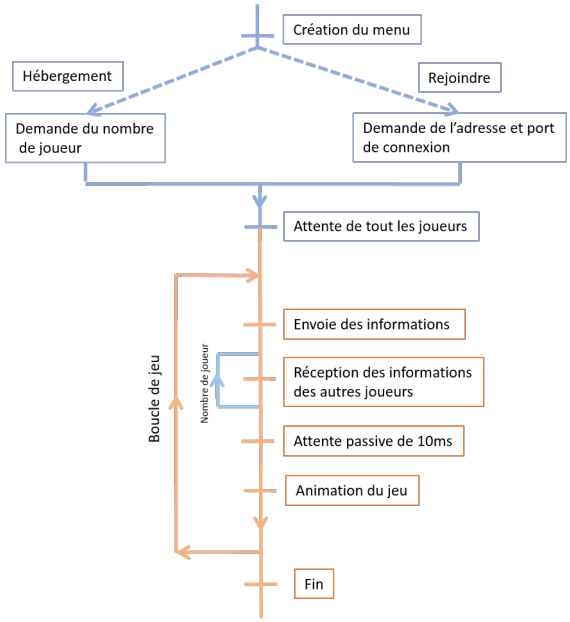

FIGURE 3.1: Déroulement du programme au cours du temps

3.1 Problèmes liés aux collisions de la balle

Nous avons rencontré quelques difficultés au niveau des collisions entre la balle et la raquette. Nous avons décidé que ce serait la balle qui gérerait les collisions avec son entourage. Si il y a une collision

(balle/raquette) alors la balle change de trajectoire (rebondit). Le rebond a été difficile à gérer car il a fallu gérer à la fois les rebonds sur les bords des raquettes et le rebond sur les différentes orientations des raquettes (horizontale/verticale) dans le cas d'un jeu à plus de 2 joueurs.

3.2 Implémentation de l'image de la raquette

Au début du projet, nous avions choisi de laisser la représentation de la raquette par une image. Mais lorsque nous avons commencé à implémenter les Bonus, si nous voulions changer la taille de la raquette, il aurait fallu créer de nouvelles images pour chaque dimension (grande ou petite) et pour chaque orientation (verticale ou horizontale) de la raquette. De ce fait, nous avons choisi de représenter notre raquette sous la forme d'un rectangle qui serait directement dessiné en fonction des hauteurs et largeurs voulues. Pour cela, nous avons utilisé la classe Drawable, qui nous permet de dessiner directement des formes sur les fenêtres.

3.3 Bonus

Nous avons rencontré plusieurs problèmes quand nous avons commencé à implémenter les bonus. En premier lieu, il a fallu choisir les conditions d'apparition du bonus et comment l'intégrer dans le protocole. Nous avions pensé au début à créer une classe pour chaque type de bonus. Mais cette solution impliquait beaucoup de modifications dans notre protocole. De ce fait, nous avons opté pour une autre solution. Nous avons directement inclu un objet de type bonus à tous les joueurs. Ce bonus apparaîtra et disparaîtra selon les besoins. Le joueur ayant perdu le 5ème point génère le bonus et le transmet aux autres. Ce système évite la triche car ce n'est pas toujours le même joueur qui génère le bonus. Lorsqu'un joueur génère un bonus, il lui donne une position, une vitesse et une direction ainsi qu'un numéro aléatoire pour définir le type de bonus (Rétrécissement/ agrandissement de la taille de la raquette, Ralentissement/accélération de la vitesse de la raquette). Nous avons fait en sorte que le cadeau rebondisse pour éviter qu'il disparaisse si il n'est pas attraper, ce qui laisse une chance à tous les joueurs.

3.4 Remise en jeu de la balle

Quand un joueur perd un point, il faut remettre la balle en jeu. Au début, nous la remettions au milieu de la fenêtre avec une vitesse et une direction aléatoire. Mais nous avons eu un problème car les joueurs se font vite surprendre par la réapparition de la balle puisque la trajectoire est aléatoire.Nous avons donc décidé de ne pas la remettre en jeu directement. Quand elle disparaît nous la replaçons à coté de la raquette du joueur qui vient de perdre. Celui-ci devra, pour la relancer, appuyer sur la touche "space" ou "espace" en français. La balle est alors remise en jeu avec une trajectoire et une vitesse aléatoire. Par contre, la balle ne suit pas la raquette du joueur.

3.5 Difficultés du passage en réseau

Une des principales difficultés a été de réaliser le projet de manière décentralisée. En effet, chaque joueur possède son propre serveur. Pour ce faire, il faut que tous les joueurs soient connectés entre eux et non à un

seul serveur. On a mis en place un protocoles de communication permettant de réaliser cette connexion cf : Figure 4.2.

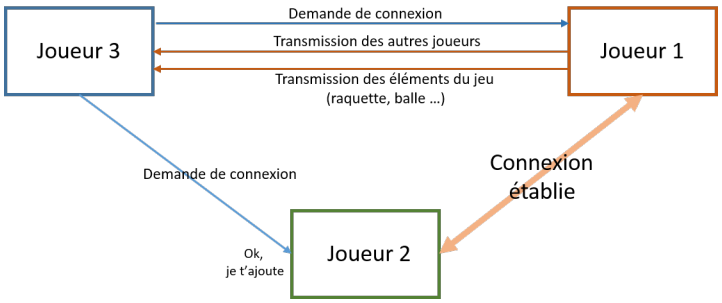

FIGURE 3.2: Procédure de Connexion

Au début nous avions réalisé le projet en utilisant seulement un *ServerSocket* mais comme il est bloquant nous avons abandonné. Nous avons finalement choisi d'utiliser un *ServerSocketChannel* permettant de faire du non bloquant. Néanmoins pour nos communications nous avons gardé une lecture bloquante avec la fonction "*socket.readline()*". Nous avons utilisé cette fonction car elle nous permet de lire le message d'un seul coup en entier. Le fait que cette fonction soit bloquante, ne gène pas énormément car nous savons que nous allons recevoir un message donc il ne bloque que très peu de temps.

Durant le jeu, chaque joueur fait les calculs des trajectoires de chacun, cela permet de vérifier que personne ne triche. A chaque tour de boucle principale le joueur envoie la position de sa raquette ainsi que la position de la balle, et son score.

3.6 Contrôle de la balle

Un seul joueur peut contrôler la balle à un instant t. Donc il a fallu mettre en place un moyen de savoir qui la contrôle. Pour cela, nous avons choisi d'utiliser les distances entre les raquettes et la balle. Pour chaque raquette i nous calculons :

$$D_i = \sqrt{(X_{balle} - X_{raquette})^2 + (Y_{balle} - Y_{raquette})^2}$$

$$D_{minimum} = min(D_0, ..., D_n)$$

La raquette dont la distance $D_i = D_{minimum}$ est la raquette maître. Tous les autres joueurs vérifieront et mettront à jour les coordonnées par rapport à lui.

3.7 Gestion des scores

Nous avons mis en place un système de score, pour permettre la compétition entre les joueurs. Il est basé sur le principe que lorsqu'un joueur manque une balle ce sont tous ses adversaires qui marque un point.

3.8 Récupération de l'adresse IP publique

Pour faciliter la connexion entre les joueurs, nous avons décidé d'afficher l'adresse de connexion et le port sur la fenêtre de connexion. Le problème est de récupérer l'adresse de connexion et pas l'adresse locale. Ne sachant que récupérer une adresse locale, nous avons du récupérer une partie de code sur internet pour écrire cette fonction. Cela fonctionne au Cremi car l'adresse rechercher est la dernière que l'on récupère grâce au code récupéré. Pour que cela fonctionne partout, il faudrait mieux récupérer son adresse publique par un autre moyen.

Chapitre 4

Diagramme de Classe du *PONG*

Ci-dessous vous trouverez deux diagramme de Classe la première contenant que les relations d'héritage entre les classes et la seconde contenant toutes les dépendances entres les classes.

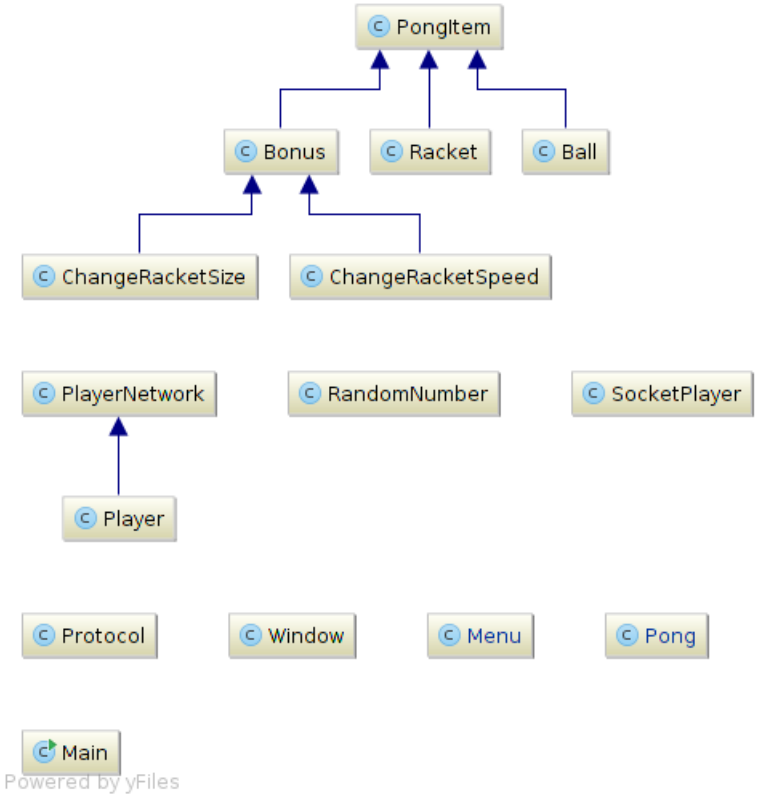

FIGURE 4.1: Diagramme de classe avec héritage

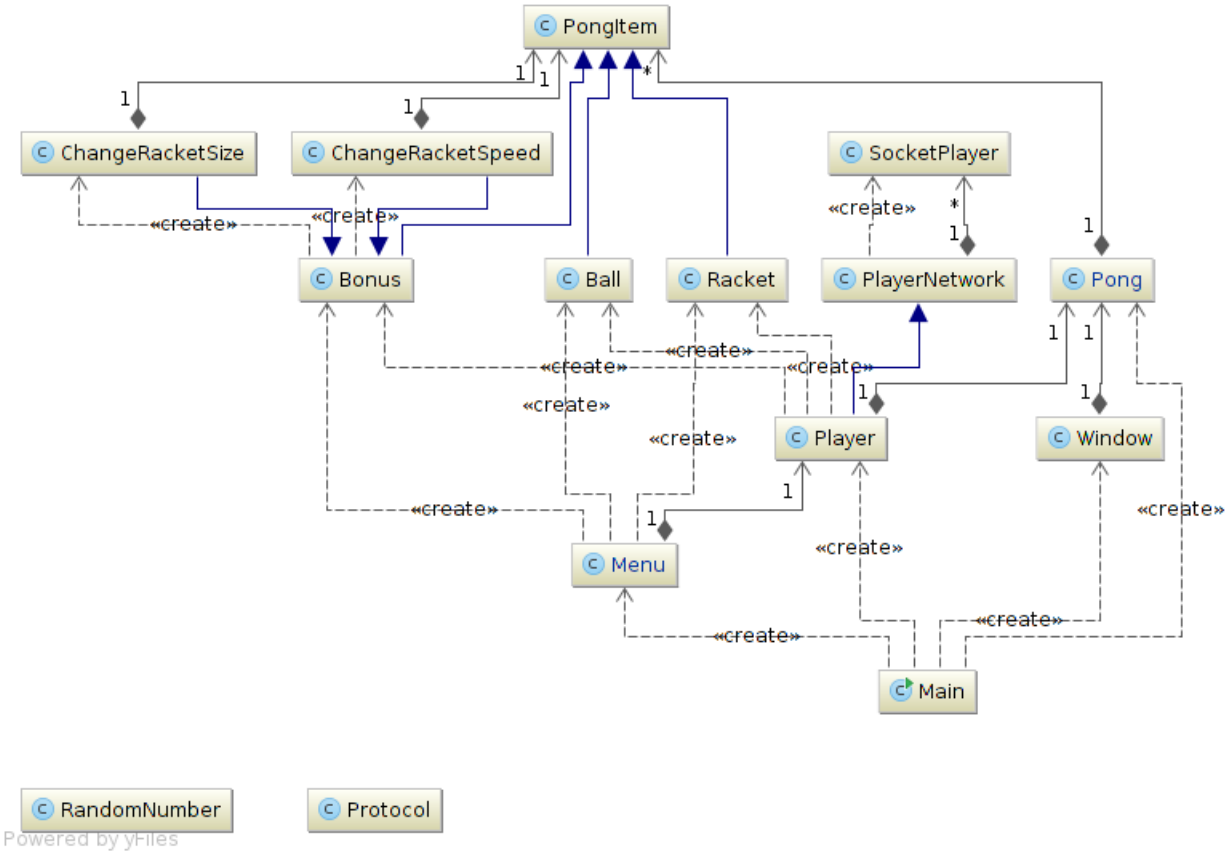

FIGURE 4.2: Dagramme de classe avec toutes les relations

Chapitre 5

Optimisation possible

Dans cette partie, nous expliquons toutes les optimisations et fonctionnalités qui pourrait être apportées à notre Pong.

5.1 Pseudo

La possibilité d'identifier les différents joueurs grâce à un pseudo. De cette manière, le pseudo pourrait être afficher dans la fenêtre de jeu ainsi que dans le tableau des scores. Cela rendrait le Pong personnalisable et permettrait d'identifier plus facilement les joueurs (gagnant et perdants).

5.2 Bonus

5.2.1 Plusieurs Bonus en même temps

Nous pourrions gérer plusieurs bonus en même temps. Dans le cas, par exemple, qu'un bonus ne soit pas attraper et que 5 points sont marqués.

5.2.2 Rocher

Pour augmenter la difficulté du jeu, nous pourrions rajouter un bonus de type "Rocher". C'est-à-dire qu'il apparaîtrait sur la fenêtre de jeu à un endroit aléatoire et le balle rebondirai dessus, la rendant plus imprévisible. Et par exemple pour le détruire, il faudrait que la balle le touche 3 fois.

5.2.3 Colle

Avec ce Bonus, nous pourrions garder la balle coller à notre raquette au lieu qu'elle rebondisse et ainsi gérer au mieux le renvoi de la balle en appuyant sur la barre espace par exemple.

5.2.4 Plusieurs balles

Afin de pimenter le jeu, le bonus pourrait nous offrir une balle supplémentaire durant un certain temps. Avec les balles qui rebondiraient les unes contre les autres.

5.2.5 Augmenter/Ralentir la vitesse de la balle

Ce bonus permettrait de diversifier un peu le rythme de jeu, et s'appliquerait évidement à toutes les balles en jeu.

5.2.6 Bonus difficile

Nous pourrions pour compliquer l'accès aux Bonus, les faire disparaître dès qu'ils touchent un bord de l'écran et les faire rebondir contre la balle (la balle aussi pourrait subir l'effet du bonus).

5.3 Mode de jeu par équipe

Nous pourrions choisir de jouer par équipes. A la place d'utiliser les quatre côtés de l'écran, nous positionnerions deux joueurs à droite par exemple et deux joueurs à gauche. Exactement comme au Ping-Pong.

5.4 Chat

Puisque nous permettons aux joueurs une pause avant de remettre la balle en jeu, nous pourrions imaginer un système de chat qui leur permettrait de dialoguer durant la partie.

5.5 Relancer

Si nous choisissions le mode de jeu avec une limite de points, nous pourrions créer un bouton relancer qui remettrait les scores à zéro et relancerait la partie.

5.6 Manche

Avec le système de relance de partie, nous pourrions créer un système de victoire par nombre de manche gagnée. A chaque victoire, le gagnant remporte une manche, le jeu se relancerait jusqu'à ce qu'un des joueurs aient gagné le nombre de manche souhaités.

5.7 Graphisme

Nous pourrions imaginer des effets et des animations pour les déplacements de raquettes, cadeaux, balles et pendant le délai d'attente de connexion de joueurs. Les graphismes de base aussi pourraient être amélioré.

Chapitre 6

Bibliographie

- OpenClassroom :

  Nous avons utilisé ce site pour apprendre à créer des interfaces graphiques, tel que les menus.

  https ://openclassrooms.com/courses/apprenez-a-programmer-en-java

  Cliquer ici

  Plus précisément :

  https ://openclassrooms.com/courses/apprenez-a-programmer-en-java/notre-premiere-fenetre

  Cliquer ici

- StackOverFlow :

  Nous avons utilisé se site pour essayer de récupérer l'adresse public. Cela ne marche pas extraordinairement bien, il est préférable de la récupérer manuellement.

  http ://stackoverflow.com/questions/32158815/how-do-you-display-the-ip-address-for-a-specific-interface
  Cliquer ici

- Documentation Oracle, API java 8 :

  Nous avons utilisé la documentation de java, pour connaître ou vérifier les prototypes des fonctions déjà implémenter par java, qui nous servent pour le réseau et interface graphique.

  http ://docs.oracle.com/javase/8/docs/api

  Cliquer ici