

Super Ballistix Ultra Deluxe Edition

Lilian CHAMPROY
Axel DUBROCA

3 janvier 2016

Table des matières

1	Préambule	2
2	Game Design	3
2.1	Crash Bash et Ballistix	3
2.2	Le Gameplay visé	4
3	Problèmes rencontrés	5
3.1	Les collisions	5
3.2	Le système de lobby	10
3.3	Les threads	15
3.4	Manque de temps	15
4	Conclusion	16

Chapitre 1

Préambule

En guise d'introduction, je souhaiterais rappeler le sujet tel qu'il est présenté et ainsi donner une vision du sujet pour y donner un sens exact.

On nous donne les indications suivantes :

- Le projet sera de réaliser un pong en réseau.

- Parmi les possibilités d'ajouts de fonctionnalités, vous pourrez proposer un jeu jouable à 4 joueurs ou de rajouter des options et divers bonus.

Ainsi, en synthétisant les idées, on arrive à un certain constat. Avec ce qui nous est proposé de réaliser, nous avons techniquement les outils pour faire un mélange entre pong et un casse-brique.

Bien que le multijoueurs à 4 ne soit pas caractéristique de pong ni d'un casse-brique, les bonus présentés (agrandissement de raquette, magnétisme, etc...) sont en revanche un trait signature de ce genre de jeu.

Je citerai en exemple de jeu utilisant ce genre de bonus Arkanoid ou Wizorb. Néanmoins, contrairement aux jeux évoqués, il n'y avait pas de multijoueurs disponibles. Par conséquent il y aura sûrement des aspects qui devront être modifiés dû aux 4 joueurs.

Ainsi, nous commencerons dans ce rapport à définir le game design de notre jeu. D'énoncer par la suite les problèmes visibles que l'on a pu rencontrer et y donner les solutions en présentant notre projet final.

Chapitre 2

Game Design

Dans un premier temps, nous évoquerons le Game design que nous souhaitons mettre en place, les idées de jeu indiquées dans le sujet ou de notre initiative. Cependant il ne serait pas juste de confirmer nos idées sans parler d'une de nos inspirations supplémentaires, à savoir : Crash Bash.

2.1 Crash Bash et Ballistix

Crash Bash est un jeu sorti sur PS1 en 2000 et que l'on pourrait qualifier de Party game.

Dans l'univers du bondissant bandicoot, toute la clique se verra s'affronter sur des mini jeux divers et variés jusqu'à 4 joueurs.

De la castagne pure à coups de caisses en bois et de dynamite, de sortir vos adversaires d'un ring à dos d'ours polaire.

Mais ici, c'est le mini jeu nommé Ballistix qui nous intéresse le plus, et qui nous aura le plus donné d'idées pour la suite.

Dans Ballistix, chaque joueurs sont placés sur les cotés de l'écran et leur but est d'éviter de perdre de la vie en prenant des balles dans leur camp, une balle dans un camp = un point de vie en moins pour le propriétaire du camp. Les billes apparaissent quant à elles des quatre coins de l'écran. Notez d'ailleurs que l'on parle ici de "billes" au pluriel, incluant que plusieurs billes peuvent se trouver à l'écran en même temps. Un joueur n'ayant plus de points de vie voit leur camp remplacé par un mur.

Les joueurs ont comme possibilités de se déplacer dans leurs cages pour défendre (sur un axe, horizontal ou vertical suivant la position), de se déplacer plus rapidement en gardant une touche enfoncée au risque d'avoir une plus grande latence pour se déplacer dans l'autre sens, ainsi que d'émettre une onde de choc pour repousser violemment (ce coup ne peut pas être utilisé indéfiniment, si le joueur l'utilise trop souvent, 2-3 fois d'affilée, le coup devra se recharger pendant plusieurs secondes avant d'être employé à nouveau).

De plus, il existe des variations de ce mini jeu.

Les variations sont les suivantes : Crashball : la version classique. Les règles sont celles citées précédemment. Les joueurs commencent avec 15 points de vie chacun.

Beachball : Pas de vague de répulsion disponible, les joueurs possèdent le bonus de magnétisme : En restant appuyé sur un bouton, les joueurs peuvent faire en sorte que les balles restent collés à la raquette. En relâchant le bouton, les balles sont propulsées, en laissant trop longtemps le bouton appuyé, le magnétisme se désactive et les balles sont relâchées très lentement. Il y a un temps d'attente avant de pouvoir réutiliser ce pouvoir. Les joueurs commencent avec 12 points de vies.

N. Ballism : De temps à autre, des bonus de répulsions apparaissent dans les coins de l'écran. Les balles sont déviées dès qu'elles approchent d'un joueur ayant ce bonus. Autre caractéristique, de temps à autre, un personnage apparait au centre de l'arène et distribue pendant un temps donné des balles dans toutes les directions, en sens horaire. Les joueurs ont 20 points de vie

Sky Balls : De temps à autres la plateforme où sont situés les joueurs penche au niveau de l'un de ses coins (décidé aléatoirement). Les balles voient leur déplacement modifié, et suivront la pente du terrain. Les joueurs ont 20 points de vie .

2.2 Le Gameplay visé

Le Gameplay visé est simple. Nous souhaitons avoir un jeu jouable de 2 à 4 joueurs reprenant les idées des jeux cités précédemment : Pong, Arkanoid et Ballistix.

Nombre de joueurs possibles : de 2 à 4.

Il sera possible de jouer de 2 à 4. A 2 joueurs, chaque joueur sera placé en face à face, sur le côté gauche et droite de la fenêtre de jeu.

A 3 joueurs, les côtés gauche et droit resteront utilisés, le troisième sera situé en bas. A 4 joueurs le quatrième est situé en haut du terrain.

But du jeu : Le but du jeu est différent suivant le mode de jeu, car nous souhaitons faire deux types de jeu.

Le premier mode de jeu serait le mode classique de pong. Les joueurs doivent marquer le plus de points en envoyant une balle dans un camp adverse.

Le deuxième mode de jeu, est le mode Ballistix de Crash Bash, les joueurs doivent éviter au maximum d'avoir des balles dans son camp. Le dernier joueur en vie gagne.

En plus de la possibilité d'avoir des modes différents, nous avons comme idée de pouvoir personnaliser notre raquette en la spécialisant.

Ce procédé se serait via le choix d'un personnage avant une partie. Ainsi, un personnage peut donner un bonus de départ qui tiendrai sur toute la partie. Magnétisme, répulsion, raquette plus grande ou plus rapide, plus de vie au départ, etc...

Le jeu aurait la possibilité de proposer un lobby qui resterait en partie.

Une fois une partie lancée, si le nombre de joueurs maximal souhaité n'est pas atteint (nombre déterminable quand on lance une partie) le joueur souhaitant rentrer dans la partie a la possibilité de rejoindre la partie en cours et d'y prendre part au recommencement d'une partie.

Chapitre 3

Problèmes rencontrés

Nous avons rencontrés énormément de problèmes. Et nous pesons nos mots. Il nous a été difficile de pouvoir avancer dans le projet sans rencontrer des impossibilités ou méthodes complexes (trop complexes?). Ces problèmes étaient parfois inhérents et intrinsèques au sujet en lui-même, plus certaines contraintes hors du contexte.

Voici donc les nombreuses difficultés que nous avons eu.

3.1 Les collisions

Les collisions, c'est de manière récurrente un véritable calvaire à gérer dans un jeu. Il y a les collisions que l'on souhaite et celles qu'on ne voulait pas...

Ici les collisions à gérer sont "simples". La présence de guillemets est très importante, car une collision n'est jamais vraiment simple en soit. Cette collision était la gestion : Balle/Raquette.

Ainsi on avait surtout une interaction de base : la balle touche la raquette, la balle rebondit.

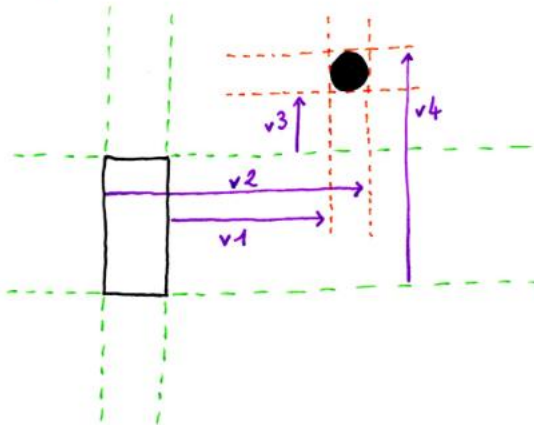
Il y avait évidemment les autres réactions via le magnétisme et la répulsion comme effet souhaités, mais il fallait avant tout se concentrer sur cette interaction de base.

Voici de nombreuses notes prises sur les interactions :

Collisions

5

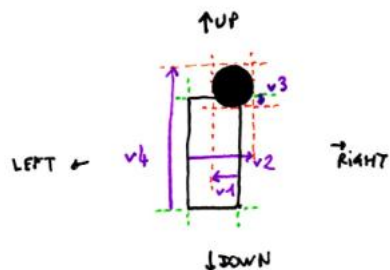
→ Balle / palette



Collision si $(\text{sens}(v1) \neq \text{sens}(v2)) \ \&\& \ \text{sens}(v3) \neq \text{sens}(v4)$



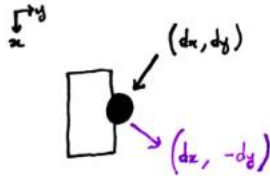
Détecter quelle face est en collision ? On prend le vecteur dont la longueur est la plus courte, et dont le sens est "vers l'intérieur". Ce n'est pas particulièrement fiable ...



$v3$ est le plus court. Son origine ? La face UP.

FIGURE 3.1

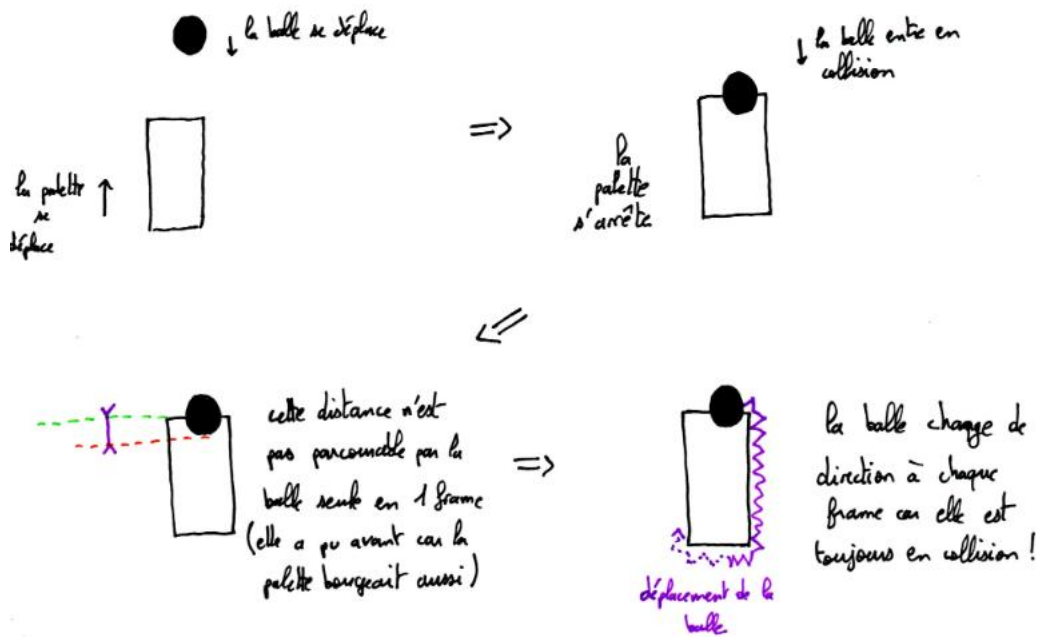
En cas de collision, selon la face, on inverse le sens de déplacement de la
balle. 6



< phase de test >

Méthode comportant un bug majeur. Ces opérations sont effectuées à chaque frame (30 fps).
10ms entre chaque

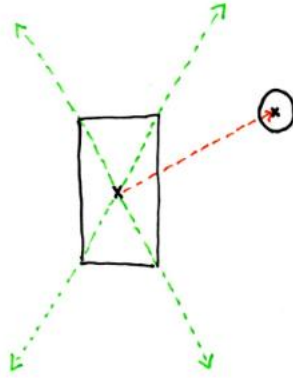
Quand la palette va dans le sens opposé de la balle, elle pénètre trop la palette :



< vidéos du bug >

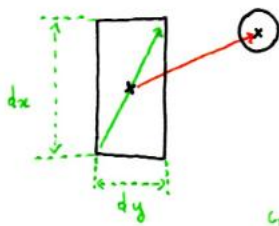
FIGURE 3.2

Pour corriger, on garde le même détecteur de collision, et pour changer la direction de la balle, on utilise d'autres vecteurs :



Selon l'angle du vecteur, la balle entre en collision avec la face correspondante.

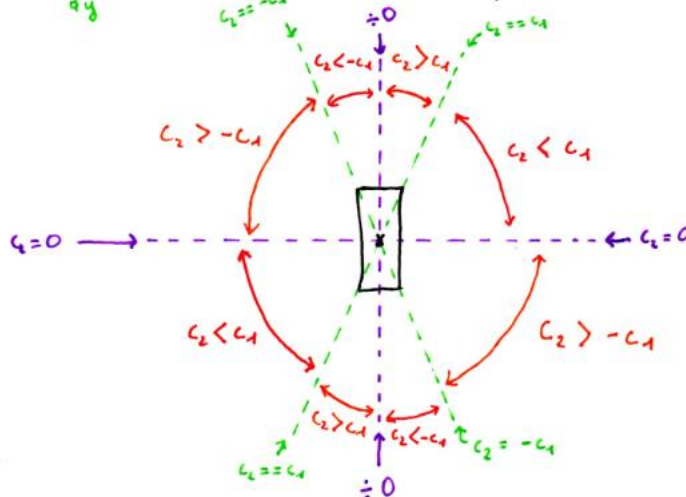
En application :



On calcule le coefficient directeur $c_1 = \left(\frac{dx}{dy}\right)$, soit le rapport de taille de la palette.

De même avec le 2^e vecteur $c_2 = \frac{x_{\text{mitoyen-balle}} - x_{\text{mitoyen-palette}}}{y_{\text{mitoyen-balle}} - y_{\text{mitoyen-palette}}}$.


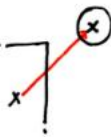
On obtient déjà 1 première information :



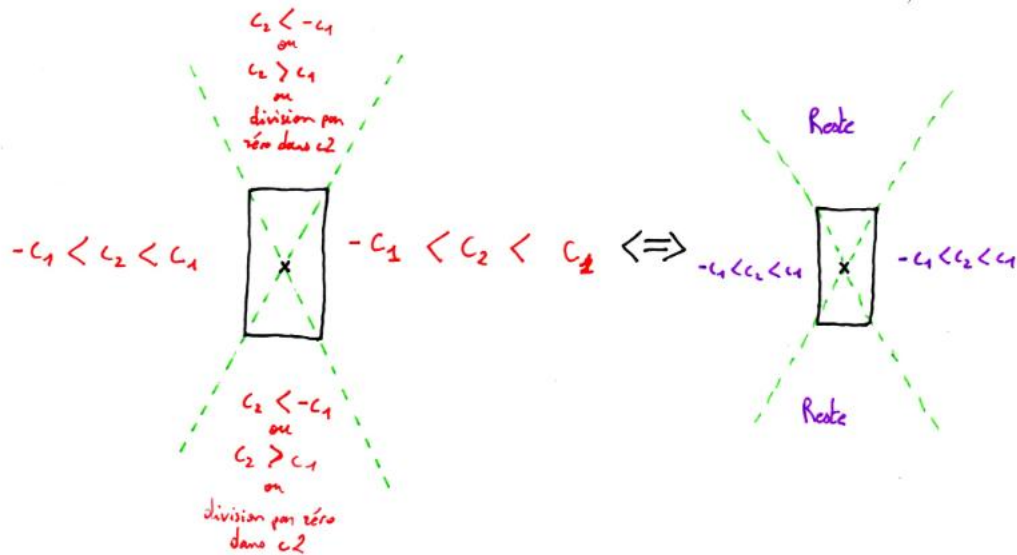
⚠ division par 0

FIGURE 3.3

Pour obtenir l'information que nous voulons (la face de collision), on n'a pas 8 besoin de toutes ces informations. On peut tout factoriser, et on obtient :

Soit  c_1 et  c_2 les coefficients

directions de ces vecteurs :




Le seul test $(-c_1 < c_2 < c_1)$ nous donne un indice sur la face de collision (UP/DOWN ou LEFT/RIGHT). Il suffit ensuite de regarder dans quelle direction pointe le vecteur .

FIGURE 3.4

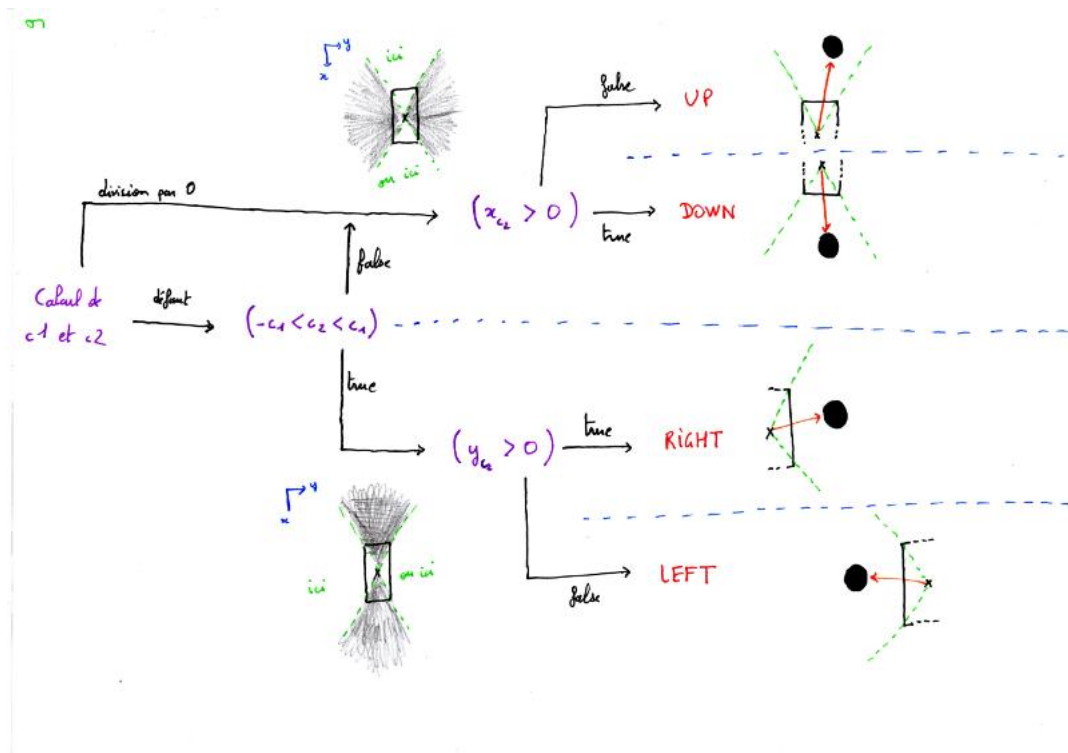


FIGURE 3.5

Problèmes liés :

-On se base non pas sur une réaction simple de rebond. Ainsi une balle qui arrive sur la raquette avec un angle α , ne rebondira pas avec un angle $2 \cdot \alpha$ dans la direction opposé. Ici nous nous basons sur une réaction avec des normales. Il y a donc plus de calculs et de vérifications, donc, potentiellement un nid de bugs en tout genre.

-Problème lié et surtout rencontré : les hitboxs. Le programme va peut être rencontrer des incohérences (deux possibilités qui sont rencontrés en même temps) et on pourra avoir une balle qui sera toujours considéré en collision ou jamais, ou autre effet indésiré.

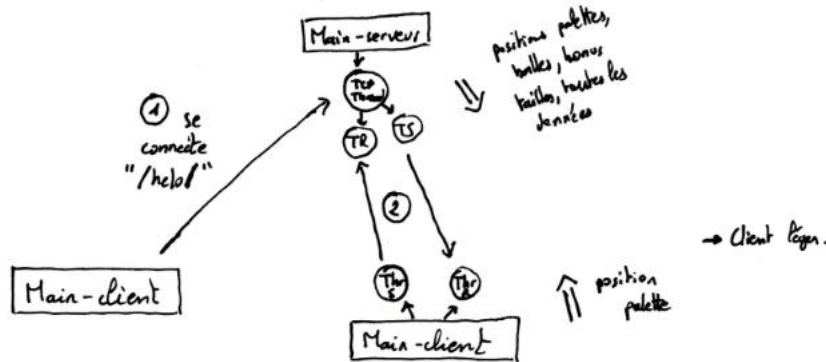
Résultat, une balle qui gravite autour de la raquette. (voir dans les annexes du git)

3.2 Le système de lobby

Nous souhaitons à la base faire un système de lobby. Un joueur créait une partie puis un autre joueur pouvait le rejoindre, et ainsi de suite pour les autres joueurs souhaitant rejoindre et ce en cours de partie également.

Voici des notes expliquant nos intentions :

Principe de base :



Le Main-serveur gère l'affichage de l'host, et le moteur physique (collisions).

Main-client ne gère que l'affichage. Les deux gèrent leurs inputs (certains objets leurs sont associés).

main envoie les positions de ses objets, et reçoit tout le reste.

Tout cela est géré par des threads.

Temps :

Main-serveur crée un Thread TCP-servem qui attend qu'un Main-client se connecte. Dès qu'une connexion est établie ("/hello/"), les deux côtés créent 2 threads : 1 pour envoyer, l'autre pour recevoir.

Ils accèdent et modifient les données en conséquence, avec des synchronized (une instance se synchronise sur elle-même :

```
public int getOwner() {
    synchronized (this) {
        return owner;
    }
}
```

1. Le protocole utilisé est symétrique : on ne peut pas dicter un message serveur à un message client. Pas très résumée. Mais c'est pas ce qu'on nous demande.

FIGURE 3.6

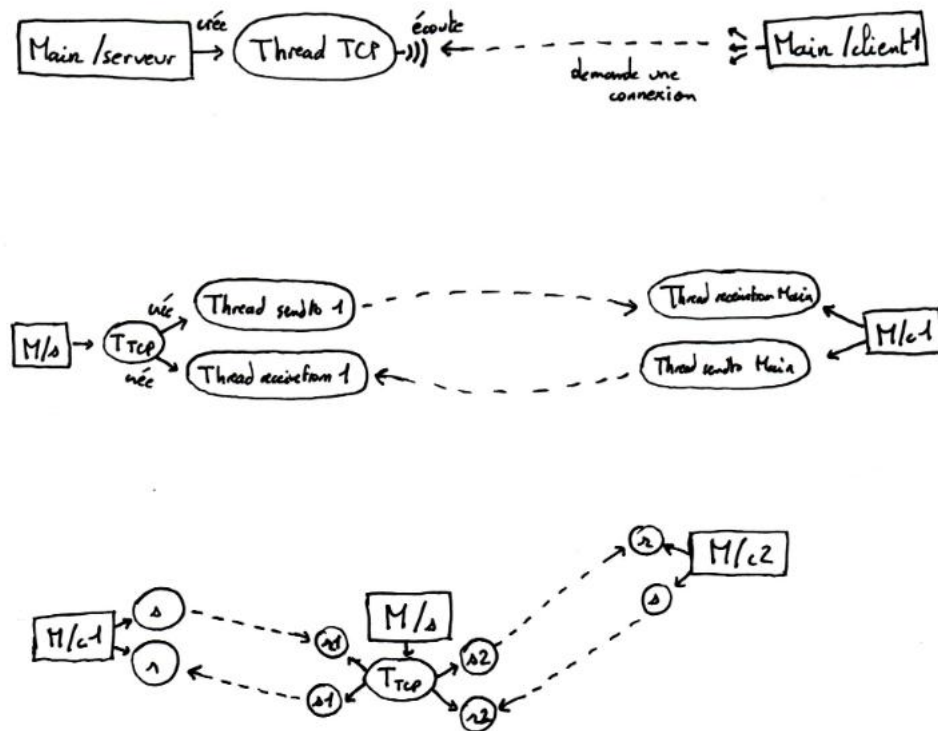
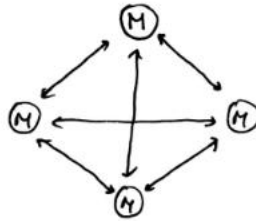
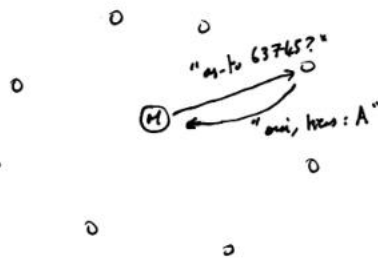


FIGURE 3.7

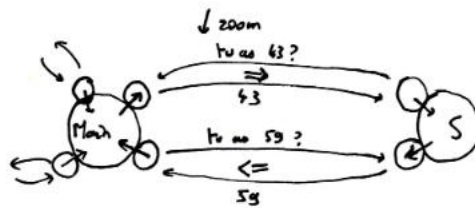
En "P2P":



- Nécessité de communiquer les adresses IP des clients.
- Tous les clients sont aussi en attente de connexion.
- Toutes les données sont envoyées ? Qui arbitre la véracité des données en cas de désynchronisation ?

P2P:

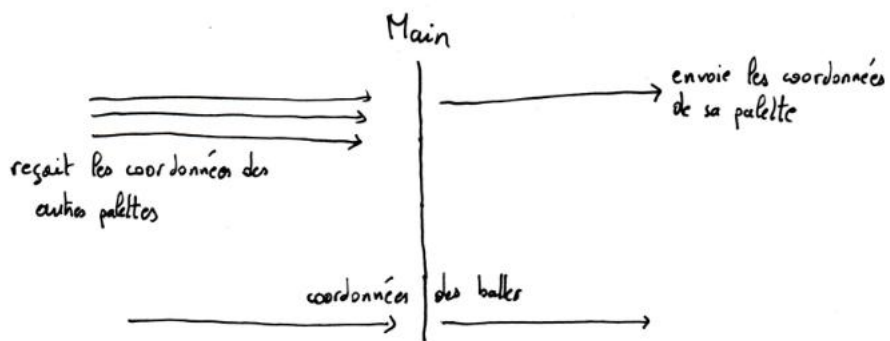
- Décompage de la donnée entre les serveurs. On demande, ils donnent.
- Pas de réelle symétrie. Chaque nœud se voit comme un client. Comme 2 threads.



- Echange, mais la symétrie s'arrête-là.
- Comme des mini-serveurs : requête, réponse.

On ne peut pas se permettre ce principe. On a une contrainte de réactivité.
 Comment attendre des clients sans thread ? Le seul moyen : phase de connexion, paquets de jeu.
 Impossible qu'un joueur s'ajoute en cours de route sans connect(), qui est bloquant. Il y a forcément des threads (ou des forks, mais ça, il en est hors de question. Si on introduit les threads, les forks aussi).
 De ce que je vois, ce cas est INSOLUBLE.

FIGURE 3.8



Qui choisit les événements aléatoires ? Il y a FORCÉMENT un centre décisionnaire.

SYMETRIE IMPOSSIBLE.

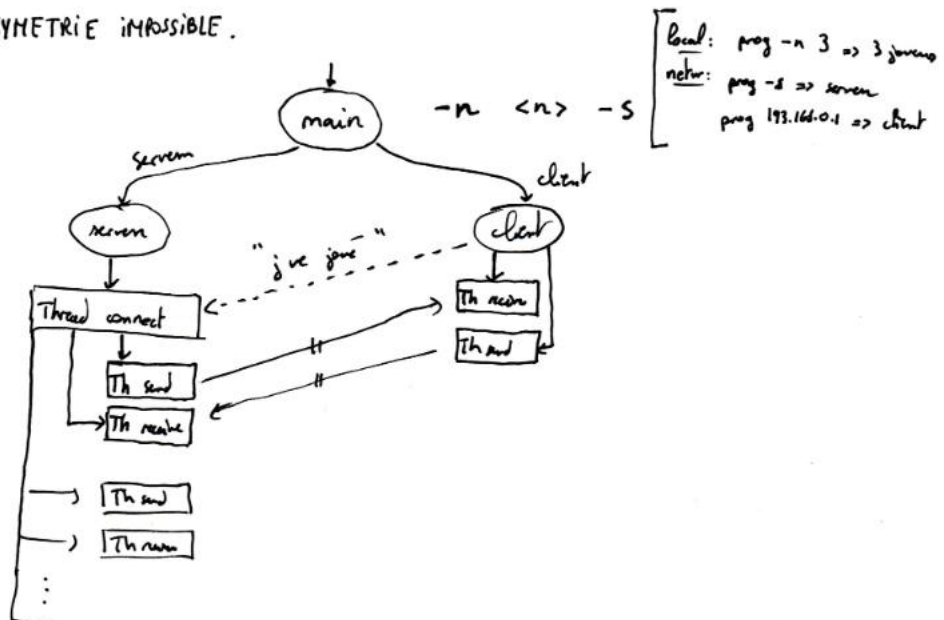


FIGURE 3.9

La également différents sources nous ont fait fausse route : -Les threads : c'est un sujet que nous évoquerons en fin de section du à son importance. Mais via ce problème, nous avons eu diverses possibilités d'actions. -Premièrement, sans les threads, nous avons eu beaucoup, beaucoup, beaucoup de mal à faire rejoindre des personnes avec notre système. D'ailleurs à partir de la première personne ayant rejoint, personne ne pouvait rejoindre la partie. De ce fait, notre système n'a pas été continué. Nous avons donc opter pour une autre solution : La partie se créait avec tous les joueurs souhaités, et impossible de rejoindre par la suite.

-Deuxièmement, nous avons senti dans un certain sens, que l'absence de GUI (graphical user interface) à limité nos possibilités de créer un système de lobby utilisable simplement et facilement pour tous. La GUI est en soi, un autre sujet que nous élaborerons dans la suite de ce rapport.

3.3 Les threads

Surement, LA, chose qui nous a entraîné vers un projet que nous ne souhaitions pas faire. On nous expliqué peu de temps après que le sujet soit donné, que l'utilisation de threads était interdite. A partir de ce constat, il nous a fallu reprendre entièrement de zéro. Toute notre structure conçu avec les threads n'était plus possible et il a fallu adapter tant bien que mal.

Et à quel prix.

Sans les threads, notre vision pour amener le jeu de manière la plus efficace s'est entièrement vue remplacé par une sorte de rustine qui va essayer de faire tenir le jeu. Et encore. Notre version actuelle ne tourne pas suffisamment correctement pour se le permettre.

Du coup, pas de mise en réseau comme nous le souhaitions, ce qui inclut également la gestion du jeu par les joueurs. Elle s'en sera vu détériorée.

Ainsi au détriment d'une vision acceptable de jeu, nous devons revoir nos ambitions à la baisse.

3.4 Manque de temps

Manque de temps pour TOUT, en réalité.

Les changements ont bouleversé nos plans initiaux. Du coup il a fallu plancher sur un projet, presque, entièrement différent.

Ainsi, n'ont pas pu être mis en place :

-la partie réseau.

-la plupart du gameplay. D'ailleurs beaucoup de points, notamment les idées des modes ou des personnages n'ont même pas pu être abordé dans nos recherches de travail...

Bref beaucoup de choses qui font que le projet n'est clairement pas abouti, et ce pour des raisons véritablement discutables.

Chapitre 4

Conclusion

Nous avons été déçus.

A la fois de ne pas pouvoir avoir quelque chose à rendre d'acceptable. Mais avant tout de ne pas avoir les outils pour le faire convenablement.

Interdiction de threads, consignes approximative dans le sujet et choix discutables.

Je vais prendre parti en tant que game designer pour vous décrire ce projet.

C'est pauvre. Pauvre en tout. Et qui plus est sur une mauvaise voie.

Que ce soit des idées ou juste d'ergonomie. C'est creux et parfois ne rime à rien.

J'en prends pour exemple la partie réseau à 4 joueurs (même si à deux c'est suffisant) :

Dans le système proposé, chacun doit gérer une partie du plateau de jeu.

Gérer l'espace de jeu pour chacun des joueurs, est déjà une aberration en soi. En plus donc d'avoir à quadriller le plateau pour définir quelle partie appartient à tel joueur, on obtient une scission claire et théorique dans le mouvement des balles.

Si une balle passe par une zone d'un autre joueur, il y aura même si l'écart est faible, un moment où la balle aura disparu de l'écran le temps que l'information du passage de la balle soit envoyé à l'autre joueur, car c'est l'autre joueur qui gère son plateau.

Prenez en compte le fait qu'il y a plusieurs balles et donc des scissions de mouvement à plusieurs endroits. Le nombre d'informations est juste trop grand, pour les interactions à faire.

Donc revoyons l'idée : des balles qui auraient la propriété de disparaître quand elle change sur d'une partie à l'autre. Très intéressant. Donc en plus d'avoir une multitude de balles, et donc de trajectoires à calculer. Il faudrait pouvoir calculer un momentum d'une balle qui ne serait plus visible l'espace d'un instant ?

Attention surprise : cette idée est complètement déraisonnable.

Alors que bizarrement, rajouter les threads et vous obtenez de suite une meilleure option, ergonomique, et viable :

Tout est géré par les threads du coup, l'hôte aurait une vision parfaite, tandis que ces adversaires auront une certaine latence. Au moins les balles seront visibles en tout points ainsi que leur trajectoires seront étudiables.

Ainsi tout les joueurs pourront avoir une expérience de jeu correct.

Comment ? si il y a trop de balles autant en réduire le nombre ? Très bonne idée, à moins que vous ne souhaitiez anéantir un point pourtant crucial.

En ne possédant qu'une balle vous aurez des parties longues et fatigantes. Et cela uniquement car il y aura beaucoup de passivité dans votre jeu et beaucoup de prédictibilité.

Mais alors pourquoi ne pas faire en sorte que la balle aille plus vite ?

Ici c'est l'accessibilité que vous détruisez. Si la balle va vite/trop vite, les petites connexions ou pc trainant la patte auront beaucoup de mal à suivre les mouvements.

C'est pourquoi notre solution était une bonne solution. Une multitude de balles, avec une vitesse convenable suffit à donner du challenge sans trop devoir forcer le joueur à prédire l'avenir ou s'impatiser pendant la partie.

Et c'est bien ici qu'est la faiblesse de ce sujet. Il n'a jamais été pensé pour être joué. Donc en soit ce n'est qu'un faux sujet croyant être attrayant en se faisant passer pour un "jeu".

Et c'est bien un regret de voir que l'ergonomie, et même ce qui pourrait passer comme étant accessoires en passant par les graphismes ou de simple features, passe au détriment d'un sujet mal décousu.

Ce que je vois donc dans ce sujet c'est savoir coder, mais en aucun cas coder un jeu.