

Ricochet Robot

OTMANE Pierre
TASSEL Joan
1A TD1

RAPPORT DE PROJET

I – Table des matières

- I - Table des matières
- II - Fonctionnalités implémentées
- III - Organisation du programme
- IV - Organisation et répartition des tâches
- V - Bilan qualitatif du travail
- VI - Difficultés rencontrées
- VII - Points parus intéressants
- VIII - Mode d'emploi
- XI - Conclusion

II - Fonctionnalités implémentées :

Le jeu est découpé en trois grands modes : Solo, Multijoueur, et éditeur.

Le mode solo charge une carte choisie par le joueur et lui laisse la possibilité de jouer ou de quitter. Si le joueur “bat” l'ancien meilleur score sur un plateau de jeu, son score remplace l'ancien score pour ce plateau.

Le mode multijoueur donne la possibilité pour 2 à 4 joueurs de jouer face à face. Chaque joueur pari sur le nombre de déplacement qu'il va faire pour faire atteindre un robot à l'objectif, celui qui arrive à son pari a gagné la manche, il y a 4 robots donc 4 manches. Le joueur ayant placé le plus de robot à gagné, si il y a égalité, tous les joueurs ont perdu.

Le mode éditeur propose au joueur de placer un robot, objectif ou mur sur la carte, en 3 étapes il peut placer un objet. Il peut également supprimer un objet, effacer totalement une carte, sauvegarder une carte ou quitter le mode éditeur.

Tous les plateaux de jeu sont sauvegardés dans des fichiers textes. Ils sont donc sauvegardés après fermeture du programme et récupérables ensuite.

La version Linux permet la saisie de touches au clavier sans la validation par la touche [entre] grâce à une fonction qui nous a été inspirée par le site : <http://forums.macg.co/developpement-mac/lire-touche-appuyer-entree-c-176764.html>

Le jeu gère la navigation dans les menus avec retour au menu principale dans toutes les étapes du jeu.

La version Linux gère la couleur pour les 4 robots.

III - Organisation du programme :

Le jeu est découpé en trois grands types de fonctions :

Les fonctions ask :

Elles permettent la demande de saisie par l'utilisateur. Tous les contacts avec l'utilisateur sont faits par ces fonctions, ainsi que la vérification des saisies.

Les fonctions display :

Elles affichent quelque chose à l'écran, le jeu, des messages ect ...

Les fonctions work :

Ces fonctions ne rendront aucun rendu à l'utilisateur, elles ne font que “travailler” et permettent de gérer tout le programme. C'est ces fonctions qui gèrent toutes les autres.

Nous avons découpé notre programme de cette façon dans le but d'avoir un code réutilisable pour une version SDL sans trop de changement, seules les fonctions display et ask changeraient.

Les fichiers ask :

ask_CommunicMap : Contient toutes les fonctions de demande de saisie utilisable par plusieurs modes de jeu.

<code>int askExchangeRobot(void)</code>

<code>int askDirection(void)</code>

<code>void askContinue(void)</code>

ask_Editor.h : Contient toutes les fonctions de demande de saisie liées au mode éditeur.

<code>int askEditorWhatToDo(void)</code>
--

<code>char askEditorChooseObject()</code>

<code>int askEditorCoordinate()</code>
--

ask_Menu.h : Contient toutes les fonctions de demande de saisie liées aux menus.

<code>int askMenuStart(void)</code>

<code>int askMenuDifficulty(void)</code>
--

<code>int askMenuEditor(void)</code>

<code>int askNumberPlayer(void)</code>
--

ask_Multi.h : Contient toutes les fonctions liées au mode de jeu multijoueur.

<code>int askBet(void)</code>

Les fichiers Display :

display_CommuneMap.h : Contient toutes les fonctions affichant des informations en rapport avec le plateau de jeu utilisables dans les différents modes de jeu.

<code>void displayMap(Object gameBoard[SIZE][SIZE], Parameters gameParameters)</code>
<code>void displayVictory(void)</code>
<code>void displayExchangeRobot(void)</code>
<code>void displayDirection(void)</code>

display_Editor.h : Contient toutes les fonctions affichant les informations liées au mode éditeur.

<code>void displayMapEditor(Object gameBoard[SIZE][SIZE])</code>
<code>void displayEditorWhatToDo()</code>
<code>void displayEditorAddWhat()</code>
<code>void displayEditorCoordinate(char letter)</code>

display_Menus.h : Contient toutes les fonctions affichant des informations liées aux menus.

<code>void displayMenuStart(void)</code>
<code>void displayMenuDifficulty(void)</code>
<code>void displayMenuEditor(void)</code>
<code>void displayNumberPlayers(void)</code>
<code>void displayHightScore(int scores[3])</code>

display_Multi.h : Contient toutes les fonctions affichant des informations liées au mode multijoueur.

<code>void displayBet(int player, char robot)</code>
<code>void displayLose(int player)</code>
<code>void displayRoundWinner(int player)</code>
<code>void displayWinner(int winner)</code>

Les fichiers Work :

work_CommunMap.h : Contient toutes les fonctions nécessaires à la gestion du plateau de jeu.

<code>void createBasicMap(Object gameBoard[SIZE][SIZE])</code>
<code>void createMap(Object gameBoard[SIZE][SIZE], Object robots[4], Parameters *gameParameters)</code>
<code>void workMove(Object gameBoard[SIZE][SIZE], Object *robot, Parameters *gameParameters, int direction)</code>
<code>void workErase(Object gameBoard[SIZE][SIZE], Object erase)</code>
<code>void workPut(Object gameBoard[SIZE][SIZE], Object add)</code>
<code>void saveMapInArray(Object gameBoard[SIZE][SIZE], Parameters *gameParameters)</code>
<code>void saveMap(Object gameBoard[SIZE][SIZE], Parameters *gameParameters, char file[20]);</code>

work_ContactFile.h : Contient toutes les fonctions qui vont se connecter à un fichier pour y lire ou enregistrer des données.

<code>void retrieveMapInformations(Parameters *gameParameters)</code>
<code>void saveMapInformations(Parameters *gameParameters)</code>
<code>void retrieveHightScore(int scores[3])</code>
<code>void saveHightScore(char scores[3], char destination[20]);</code>

work_Editor.h : Contient toutes les fonctions nécessaires au mode édition.

<code>void workEditor(Object gameBoard[SIZE][SIZE], Object robots[4], Parameters *gameParameters)</code>
<code>int workEditorAlreadyOne(Object gameBoard[SIZE][SIZE], Object object)</code>
<code>void workEditorAdd(Object gameBoard[SIZE][SIZE], Object robots[4])</code>

work_Environnement.h : Contient toutes les fonctions « Spéciales » comme la conversion, l'aléatoire, le saut de lignes ect...

<code>void environnementClearScreen(void)</code>
<code>char environnementMyGetchar(void)</code>
<code>char environnementGetCharEnterLess(void)</code>
<code>void newLine(int numberNewline)</code>
<code>void space(int numberSpace)</code>
<code>void workWait()</code>
<code>int randomInt(int iMin, int iMax)</code>
<code>int convertCharNumbToInt(char nb[3])</code>
<code>void convertIntNumbToChar(int nb, char number[3])</code>

work_Menus.h : Contient toutes les fonctions nécessaire à la gestion des menus.

<code>void menuStart(Object gameBoard[SIZE][SIZE], Object robots[4], Parameters *gameParameters)</code>
<code>void menuDifficulty(Object gameBoard[SIZE][SIZE], Object robots[4], Parameters *gameParameters)</code>
<code>void menuEditor(Object gameBoard[SIZE][SIZE], Object robots[4], Parameters *gameParameters, int edit)</code>
<code>void menuScores(int scores[3])</code>

work_Multi.h : Contient toutes les fonctions nécessaires au mode multijoueur.

<code>void workPlayMulti(Object gameBoard[SIZE][SIZE], Object robots[4], Parameters *gameParameters)</code>
<code>void workOrderPlayers(Multi players[], int numberPlayers)</code>
<code>void workCopyMap(Object gameBoard[SIZE][SIZE], Object gameBoardCopy[SIZE][SIZE])</code>
<code>int workFoundWinner(Multi players[], int numberPlayers)</code>

work_OnePlayer.h : Contient toutes les fonctions nécessaires au mode solo.

<code>void workPlaySolo(Object gameBoard[SIZE][SIZE], Object robots[4], Parameters *GameParameters)</code>
<code>void hightScore(Parameters *gameParameters)</code>

work_Random.h : Contient toutes les fontions permettant la création de maps aléatoires.

<code>void workCreateRandomMap(Object gameBoard[SIZE][SIZE], Object robots[4], Parameters *gameParameters)</code>
<code>void workRandomUpWalls(Object gameBoard[SIZE][SIZE], int numberWalls);</code>
<code>void workRandomUpObjects(Object gameBoard[SIZE][SIZE], Object robots[4])</code>
<code>int workTryMap(Object gameBoard[SIZE][SIZE], Object robots[4], Parameters *gameParameters)</code>
<code>void workRandomMoves(Object gameBoard[SIZE][SIZE], Object *robot, Parameters *gameParameters, int direction)</code>

IV - Organisation et répartition des tâches

On beaucoup travaillé à deux sur un seul ordinateur pour coder le projet et déboguer. En dehors des heures de travaux en commun, on avançait chacun de notre côté sur les mêmes fonctions du projet pour avoir différentes idées possibles et on choisissait la meilleur pour la mettre dans la version correcte du jeu.

V - Bilan qualitatif du travail

Le projet à subit 4 grosses versions.

Pour la première version, on ne savait pas trop où on allait, on a essayé de faire déplacer les robots sur le plateau et on a continuer. On a fini par avoir un mode solo jouable, cependant le code était trop lourd, mal réparti. Lorsque nous avons découvert le principe de structures, on a recommencé le projet en reprenant les idées intéressantes de l'ancienne version.

Pour la seconde version, on à réussi à rendre le jeu jouable avec le mode solo, multi et éditeur, environ 2 mois avant de rendre le projet. C'est à ce moment que l'on s'est tourné sur l'adaptation en SDL du projet.

Pour la troisième version, on s'est attaqué à la SDL, après deux semaines, on à réussi à avoir un jeu jouable solo en version graphique. Cependant, le code source était presque totalement différent du la version console, c'est là qu'on à compris que notre code n'était pas adaptable.

Enfin, un mois avant de rendre le projet, on s'est rendu compte que la version console n'était pas optimisée ni terminée à 100 %, on à cherché à la terminer. Cependant on s'est vraiment rendus compte que le code était mauvais, mal organisé, manquait de fonction, trop lourd etc...

On à décidé de recommencer à 0 une version console en ayant toujours en tête une adaptation « rapide » vers une version graphique. C'est là qu'on à eu l'idée de découper le code en 3 grands types de fonctions : ask, display et work. On en a profité aussi pour coder le jeu totalement en anglais.

Aujourd'hui, le code source nous plaît, il reste, certes quelques imperfections comme des fonctions qui pourraient être plus modulaires.

Nous comptons rendre uniquement une version console du projet, mais nous allons continuer le projet après l'avoir rendus pour faire la version graphique qui nous tiens à cœur.

VI - Difficultés rencontrées

Au final, on a eu des difficultés au niveau de la répartition du travail, nous n'avons pas le même niveau en programmation, il est donc un peu difficile de coder ensemble, placer des conventions au niveau du code etc ...

Les parties du projet qui nous on posé problème étaient la création de map aléatoire, car pour le déboguage, nous avons eu pas mal de problème, nous avons été obligé d'afficher tout ce qui devait ce passer à l'écran .

L'autre partie qui nous à posé problème est le mode multijoueur, car il demande de jongler avec beaucoup de variables. Nous pensons que le code de notre fonction multijoueur n'est pas encore parfait.

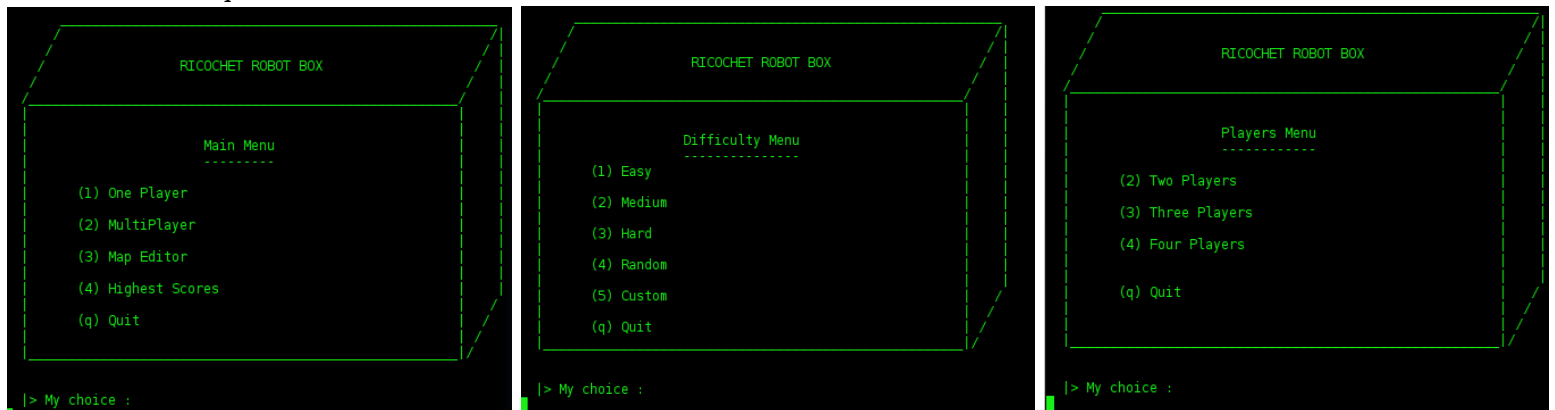
VII - Points parus intéressants

Les points qui nous ont paru intéressants étaient l'éditeur, qui permet de pouvoir modifier une map de façon ludique et rapide. La map aléatoire nous a paru aussi très intéressante d'un point de vue technique et de l'utilisation du random.

VIII - Mode d'emploi

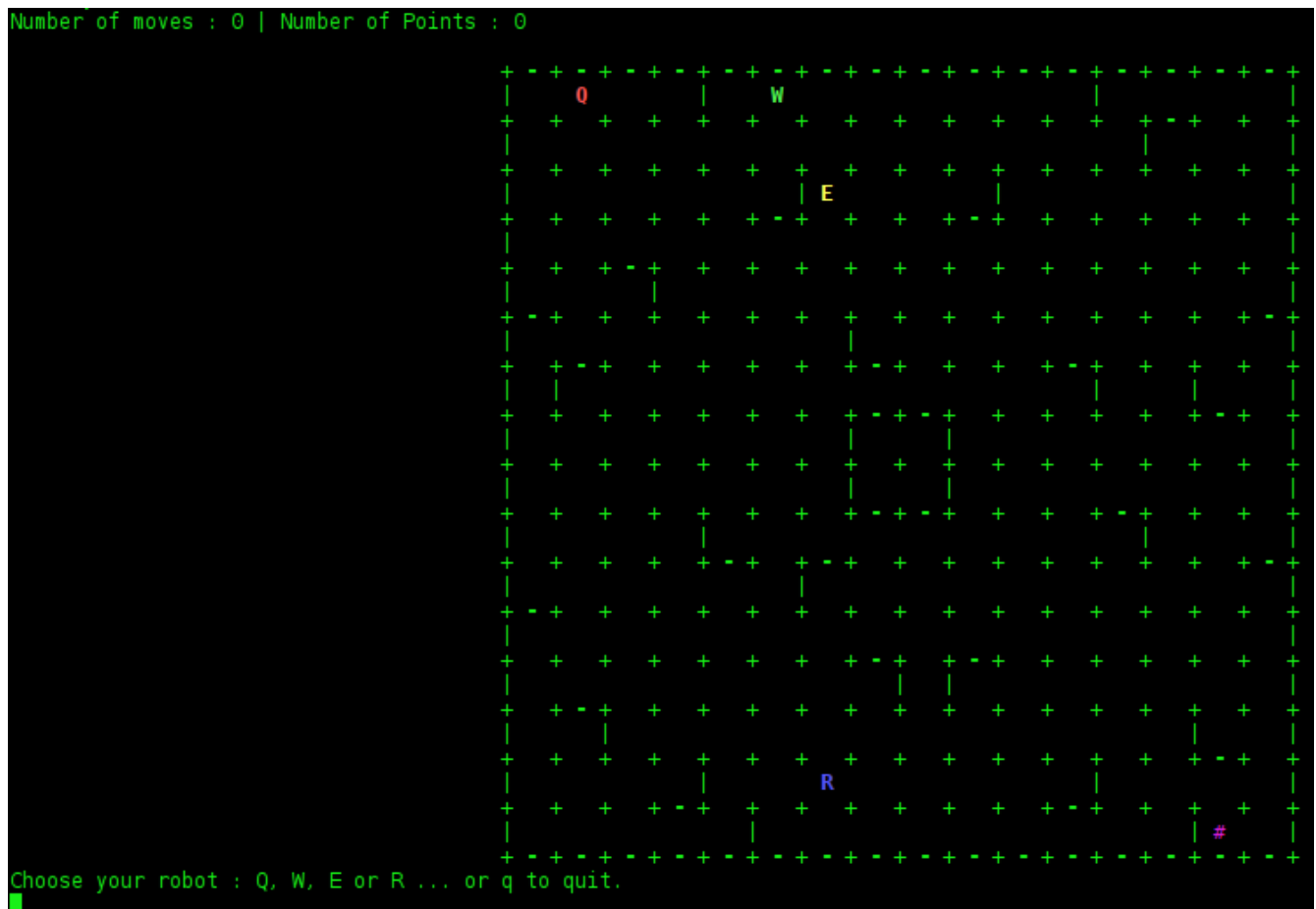
1) Menus

On navigue dans les menus avec des chiffres.



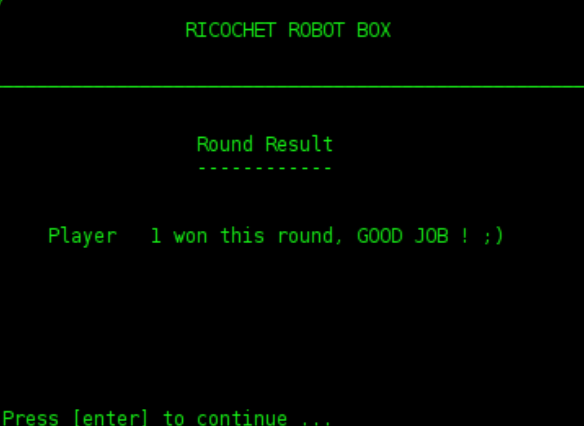
2) Mode Solo

Le mode solo permet de déplacer les robots sur le terrain.




```
Player 1 bet on the number of moving it will do to move the robot W. (Or 'q' to quit)
My bet : █
```

```
Actual player : 2
Number of moves : 6 | Bet : 15 | Robot to place : W | Number of robots to validate : 4
```



```

RICOCHET ROBOT BOX

Round Result
-----

Player 1 won this round, GOOD JOB ! ;)

Press [enter] to continue ...

```

```

RICOCHET ROBOT BOX

Round Result
-----

Player 1 lose ... ..

Press [enter] to continue ...

```

10

3) Mode Editeur

```

+ 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
01| + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - +
02| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
03| + # + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
04| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
05| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
06| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
07| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
08| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
09| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
10| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
11| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
12| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
13| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
14| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
15| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
16| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
17| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
18| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
19| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
20| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
21| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
22| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
23| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
24| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
25| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
26| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
27| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
28| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
29| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
30| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
31| + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
+ - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - + - +
Press : A to Add | D to Delete | C to Clear map | S to Save map and test it | q to Quit
My choice :
```

Le joueur choisit tout d'abord ce qu'il veut faire

```

Press : A to Add | D to Delete | C to Clear map | S to Save map and test it | q to Quit
My choice :
```

A pour ajouter :

```

Add what ?
Robot (Q W E R) : (ex : R)
Goal           : 0
Wall           : M
My choice :
```

```

My choice :
0 What Coordinates ?
x : 21
  What Coordinates ?
y : 11
```

D pour supprimer

C pour effacer totalement la carte

S pour sauvegarder et tester

q pour quitter

**Notes :*

Tous les robots et objectif ne peuvent être placés qu'une seule fois. Et ont des coordonnées impaires (x est impair et y aussi)

Tous les murs ont des coordonnées à parité opposée (si x est pair, y est impair, si x est impair, y est pair)

Lors de l'ajout de mur, il n'y a pas besoin de préciser si c'est un mur vertical ou horizontal, le jeu se charge de le placer en fonctions des coordonnées.

4) Les scores

Le jeu va sauvegarder automatiquement les meilleurs scores. Par défaut tous les scores sont à 999, à vous de faire mieux.



IX – Conclusion

Ce projet était très intéressant, de par le fait que ce soit un jeu, cela nous à, je pense, motivé d'avantage. De part le fait d'avoir recommencer de multiples fois le projet à 0, on à compris qu'il était très dur de créer les bases d'un programme « ambitieux ». On à quand même fini par trouver un moyen pour ne pas se perdre dans les lignes de codes et dans la masse.

On à appris à maîtriser pas mal de notions de C, cependant, on se rend bien compte que le code source pourrait être grandement amélioré à bien des égards et surtout par de nouvelles connaissances avancées en C.

Au final, ce projet nous à fait énormément travaillé et en échange, il nous à fait apprendre beaucoup.

D'un point de vue scientifique, on se posait souvent la question de « est-ce que cet algorithme est la solution la plus rapide en terme de calculs ? » mais nous n'avions presque jamais la réponse. Il est vrai que sur un projet comme celui-ci, les processeurs d'aujourd'hui n'ont pas de problème à suivre, mais quand nous avons créer notre version SDL du jeu, nous avons eu des problèmes de lag liés au rechargement trop fréquent d'images.

D'un point de vue humain, on s'est rendu compte que développer en équipe est une tâche difficile, pour plusieurs raisons. Tout d'abord, nous n'avions jamais développé de programmes aussi ambitieux, alors pour se répartir les tâches, ce n'était pas facile. Ensuite, on remarque que notre niveau en programmation est peu élevé et que nous ne sommes pas du même niveau, il est assez dur de travailler chacun de son coté sur des fonctions utiles pour tout le programme.