

Le Mans Université
Licence Informatique *2ème année*
Conduite de projets
En cours de réparation

Geoffrey POSÉ Maël VASSENET Pierre ROCHER

2021

Table des matières

1	Introduction	3
2	Conception	3
2.1	Analyse : mise en place de l'idée	3
2.2	Cahier des charges	3
3	Organisation de travail	4
3.1	Gantt	4
3.2	Outis	5
3.2.1	SDL2	5
3.2.2	Discord	5
3.2.3	Git	5
3.2.4	Doxygen	6
3.2.5	Google Workspace	6
3.2.6	Makefile	7
3.2.7	Code erreur	7
4	Développement	8
4.1	Robot	8
4.2	Lasers	8
4.3	Map de base	10
4.4	Menu	11
4.5	Ajout des textures	11
5	Conclusion	12
5.1	Limitations	12
5.2	Améliorations	12
5.3	Bilan	12
6	Annexe	13

1 Introduction

Notre projet de fin d'année consiste à créer un jeu vidéo, après plusieurs esquisses de jeux, nous sommes parvenus à une valeur sûre, un jeu d'énigme à résoudre, niveau par niveau. Chaque niveau se compose de murs, de miroirs, d'un laser, d'une source de lumière et d'un point d'arrivée. Le but de l'énigme est de faire réfléchir un laser sur plusieurs miroirs pour alimenter la pièce en électricité et ainsi de rallumer la lumière. L'idée étant trop basique, déjà faite à de multiples reprises, nous avons décidé d'y ajouter un robot contrôlable qui déplacerait l'angle des miroirs pour compléter le parcours, et qui doit aussi éviter de croiser la trajectoire du laser sous peine d'être vaporisé et de devoir recommencer le niveau. Le tout serait en vue du dessus à la Bomberman. Le jeu ne demanderait ni doigté, ni réflexe, mais seulement de la réflexion.

Nous avons développé le projet à 3 en parallèle en répartissant les tâches, sachant que nous n'étions pas tous sur le même OS, la compilation était différente et les makefiles aussi.

2 Conception

2.1 Analyse : mise en place de l'idée

Pour mettre à bien notre projet, il nous fallait développer une map de test, composée de différents objets à coder, dont : des murs infranchissables, des miroirs rotatifs devant être des objets réfléchissants, un robot contrôlable devant se mouvoir entre les murs et un laser s'étendant à l'infini ne pouvant traverser les murs, et se réfléchit sur les miroirs afin de toucher la surface de zone de fin.

2.2 Cahier des charges

Plusieurs points nous semblaient importants au sein de ce projet, tout d'abord il n'était pas question de rejouabilité extrême mais d'un perfectionnement de score / temps. Pour incarner le rôle de jouet dans notre programme nous avons pensé à un robot soucieux de ce qu'il se passe (cf introduction). Dans le but de rajouter des difficultés il était question d'un laser devant illuminer la zone de fin de niveau, celle-ci serait obstruée par des obstacles programmés en amont.

Il est donc question d'un niveau ayant un début et une fin, où le robot devrait se balader dans le niveau afin de pivoter des miroirs et faire de son mieux pour résoudre le puzzle qui s'offre à lui.

Par conséquent, les outils nécessaires à la réalisation de cette tâche seraient des bibliothèques sous C et SDL afin de conceptualiser une fenêtre, afficher des images et gérer des inputs (clavier, souris, manette) pour progresser dans le jeu.

3 Organisation de travail

3.1 Gantt

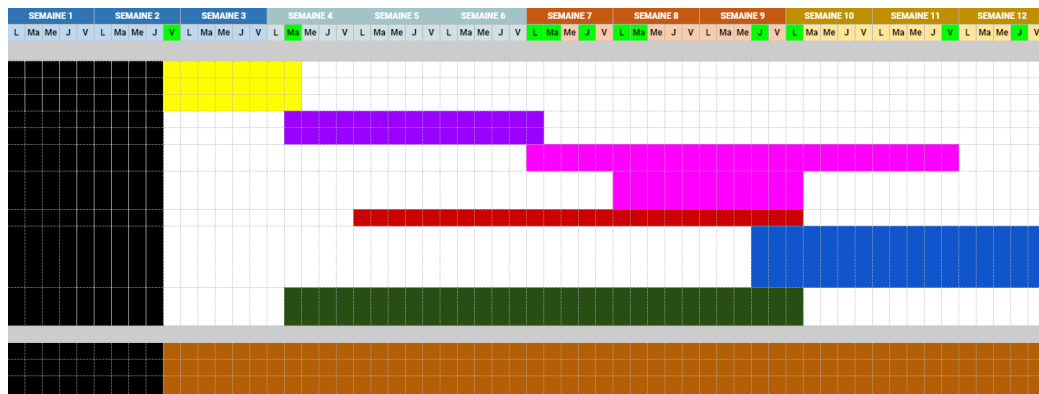


FIGURE 1 – Gantt prévisionnel

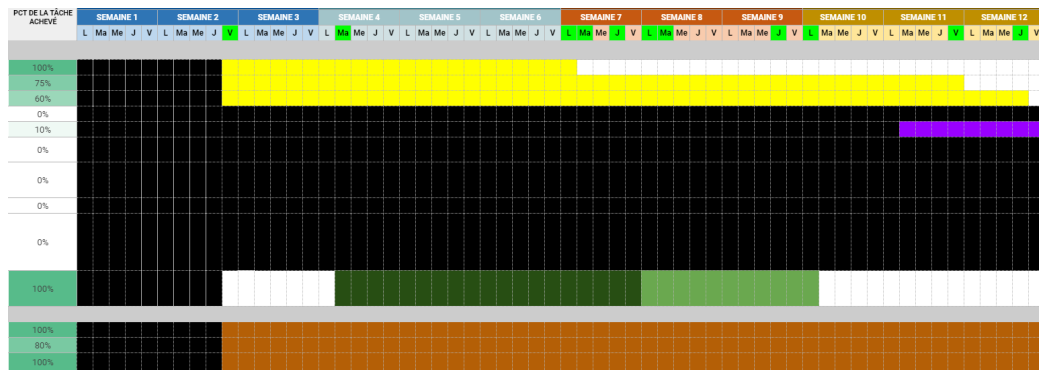


FIGURE 2 – Gantt réel

La partie en jaune représente l'avancement de la création du robot ainsi que du laser, tâche que nous avons sous-estimée et n'est à ce jour pas encore perfectionnée notamment à propos des lasers.

La partie en mauve représente le contact robot/laser, ainsi dans l'ordre "robot qui meurt au contact du laser" et "poser/pivoter des miroirs avec le

robot”. Ceux-ci n’étant pas encore finalisés, cette tâche n’a été que récemment entamée et n’en est qu’à ses débuts.

Les parties en rose, rouge et bleu ont été respectivement abandonnées consistant en la création d’un scoreboard ainsi que d’un éditeur de niveaux. Mais comme stipulé au-dessus ces tâches n’étaient que fioritures comparées à la base même du jeu qui était encore à perfectionner.

3.2 Outis

3.2.1 SDL2

Selon la restriction d’origine consistant à n’utiliser que du langage C pour le projet, ainsi que notre désir d’avoir une vraie interface graphique, il nous fallait donc utiliser une bibliothèque qui permettent de faire tout cela. Le premier choix qui nous est apparu et celui que l’on a conservé n’est autre que la SDL version 2. Celle-ci permet d’inclure facilement des textures en format BMP et permet également de faire des rendus très simplement en plus de les recharger à une fréquence précise dans le but de créer des fenêtres.

3.2.2 Discord

Il a été également mis en place un groupe de travail discord. Cela nous permet de communiquer instantanément des disponibilités de chacun, de pouvoir travailler en même temps ensemble et sans avoir la contrainte de se déplacer grâce au vocal. Nous nous en sommes servi pour se partager des ressources, comme des wikis et des morceaux de code grâce à la mise en page qu’il permet. De plus, il nous était utile lors de nos sessions de travail groupé pour voir rapidement à l’écran des autres membres du groupe pour l’aider ou simplement voir le résultat de son travail.

3.2.3 Git

Étant donné que nous travaillons à plusieurs, nous avons la contrainte de devoir partager notre code et de pouvoir fournir facilement et rapidement les dernières versions aux autres. Le tout en récupérant les versions antérieures au besoin, pour ceci il était nécessaire d’utiliser un outil de versionning, le plus populaire et le plus simple étant Github basé sur git. Cela nous permet d’avancer en équipe grâce aux différentes branches du projet, de plus nous pouvons travailler en ayant toutes les ressources des autres à disposition, que ce soit des images ou bien des sprites. Ceci permet de garder un oeil sur l’avancement général du projet afin de pouvoir se dépanner rapidement

3.2.4 Doxygen

Concernant la partie Doxygen, elle a été possible grâce à un complément de module doxygen inclus dans cygwin. Il a été explicité tout type de commentaire concernant les différentes fonctions, le tout en français. Le site est organisé de façon à pouvoir cliquer sur différentes fonctions et être redirigé vers leurs commentaires respectifs.



FIGURE 3 – Image représentative du site

Une description détaillée accompagne par ailleurs chaque fichier, daté des dernières modifications.

3.2.5 Google Workspace

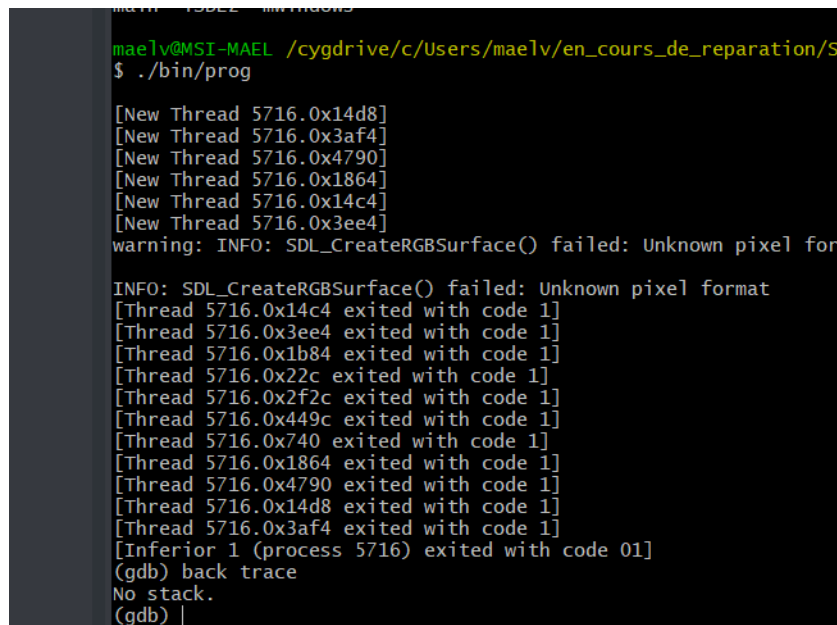
Lors de notre première séance sur le projet, nous avons été dirigés sur Google Sheets pour l'élaboration de notre Gantt. Il nous a été naturel de continuer avec les services Google avec Docs pour poser nos idées et mettre en place une trace écrite commune sur laquelle chacun d'entre nous peut modifier à sa guise. Concernant la rédaction du rapport de projet, il nous a été évident que l'on écrive sur un Docs. Qui par la suite, sera retranscrit en LaTeX pour répondre aux consignes. De plus, nous avons fait le choix d'élaborer le diaporama de notre présentation orale avec Google Slides et non en LaTeX pour pouvoir le modifier simultanément.

3.2.6 Makefile

Makefile est primordial lorsqu'on travaille en collaboration pour ainsi programmer sur une fonction bien précise sur un autre fichier autre que celle principale qui est en l'occurrence "main.c". Ce fut sans doute notre première grande difficulté puisque nous n'avions aucune compréhension de l'outil. Alors avec l'aide de notre chargé de TP, nous avons mis en place un modèle de Makefile compilable sous windows. Suite à cette aide précieuse et l'éclaircissement de l'outil, nous avons suivi le modèle pour notre jeu et par conséquent, pouvoir mettre en commun nos travaux. Néanmoins nous avons été confrontés à un autre souci, nous ne sommes pas sur le même OS, l'un de nous est sur linux, et les deux autres sur windows. Alors, il a été nécessaire d'en faire un compilable sous linux.

3.2.7 Code erreur

Nous avons mis en plus quelques essais de gdb afin de voir d'où viendraient certains problèmes qui seraient trop compliqués à repérer et donc à corriger, dont l'un lors d'un test de coloration :



```
maelv@MSI-MAEL /cygdrive/c/Users/maelv/en_cours_de_reparation/S
$ ./bin/prog

[New Thread 5716.0x14d8]
[New Thread 5716.0x3af4]
[New Thread 5716.0x4790]
[New Thread 5716.0x1864]
[New Thread 5716.0x14c4]
[New Thread 5716.0x3ee4]
warning: INFO: SDL_CreateRGBSurface() failed: Unknown pixel for
INFO: SDL_CreateRGBSurface() failed: Unknown pixel format
[Thread 5716.0x14c4 exited with code 1]
[Thread 5716.0x3ee4 exited with code 1]
[Thread 5716.0x1b84 exited with code 1]
[Thread 5716.0x22c exited with code 1]
[Thread 5716.0x2f2c exited with code 1]
[Thread 5716.0x449c exited with code 1]
[Thread 5716.0x740 exited with code 1]
[Thread 5716.0x1864 exited with code 1]
[Thread 5716.0x4790 exited with code 1]
[Thread 5716.0x14d8 exited with code 1]
[Thread 5716.0x3af4 exited with code 1]
[Inferior 1 (process 5716) exited with code 01]
(gdb) back trace
No stack.
(gdb) |
```

FIGURE 4 – Utilisation de GDB

De plus, nous avons créé une fonction de conceptualisation d'erreurs : *SDL_ExitWithError()*. Elle nous permet de savoir lorsque le programme s'exécute si une texture ne s'ajoute pas au rendu, si le téléchargement d'une

image ne se fait pas correctement. De surcroît, le message d’erreur nous permet de commenter notre code.

4 Développement

4.1 Robot

Selon nos besoins, notre robot devait se déplacer et interagir avec les miroirs pour pouvoir les orienter. Pour cela, il a été nécessaire de créer un rectangle dans lequel on y insère notre image “morty_robot” et qui fait les dimensions de cette dernière, c’est-à-dire 34x34 pixels.



FIGURE 5 – morty_robot

Le fait de mettre notre sprite dans un rectangle nous facilitait la tâche pour le faire se mouvoir. Ainsi ce n’est pas l’image que l’on déplace mais bel et bien le rectangle. Pour parvenir à ce résultat, nous étions dans l’attente d’un événement, à savoir les flèches directionnelles du clavier. Alors nous avons fait une fonction “déplacer()” qui n’est autre qu’un “switch()” dans lequel chaque “case” correspond à une flèche : gauche, droite, haut, bas. Ainsi, il nous suffisait de changer les coordonnées en abscisse notée “x” du rectangle en lui incrémentant ou décrémentant 10 (en pixels) pour les flèches gauche et droite. Et il était de même pour haut et bas pour lesquelles on modifiait les coordonnées en ordonnée noté “y”.

De plus, pour ne pas que le personnage “s’échappe” de la fenêtre, nous y avons inséré une condition pour chaque flèche directionnelle, celles-ci consistant à vérifier si le personnage se retrouve à une extrémité de la fenêtre, auquel cas il ne peut se mouvoir dans cette direction.

4.2 Lasers

La principale difficulté des miroirs est liée au fonctionnement direct des lignes en SDL. Il est nécessaire de donner les coordonnées x et y du point d’arrivée et du point de départ du trait. Cependant le laser est censé être infini, dans les faits il se limite aux bordures de l’écran, mais il est impossible de tracer un trait sans connaître au préalable le départ et la fin de celui-ci. Pour cela nous avons dû utiliser une fonction annexe qui simule le trajet du

miroir pour définir le premier obstacle rencontré, c'est-à-dire : soit le mur, le robot, ou bien le point d'arrivée.

Lorsque le laser rencontre un miroir il doit changer sa direction, en code cela se traduit par la fin du premier trait, et le calcul du second ayant comme origine la fin du premier et comme angle d'incidence 90° .

Le rayon laser se réfléchissant à 90° sur le miroir en ligne droite, il est donc nécessaire de passer par une fonction affine du second degré de la forme $ax+b$ pour pouvoir tracer le trait indéfiniment.

Avec comme constante a qui vaut l'angle d'incidence, b vaut les coordonnées d'origine et x les coordonnées à chaque itération de la fonction. Le rayon ne doit stopper sa course qu'à deux conditions : rencontrer un mur conclut sa poursuite, s'il rencontre un miroir il doit se réfléchir à 90° dessus afin de continuer sa course, ou bien rencontrer l'interrupteur final ce qui termine le niveau.

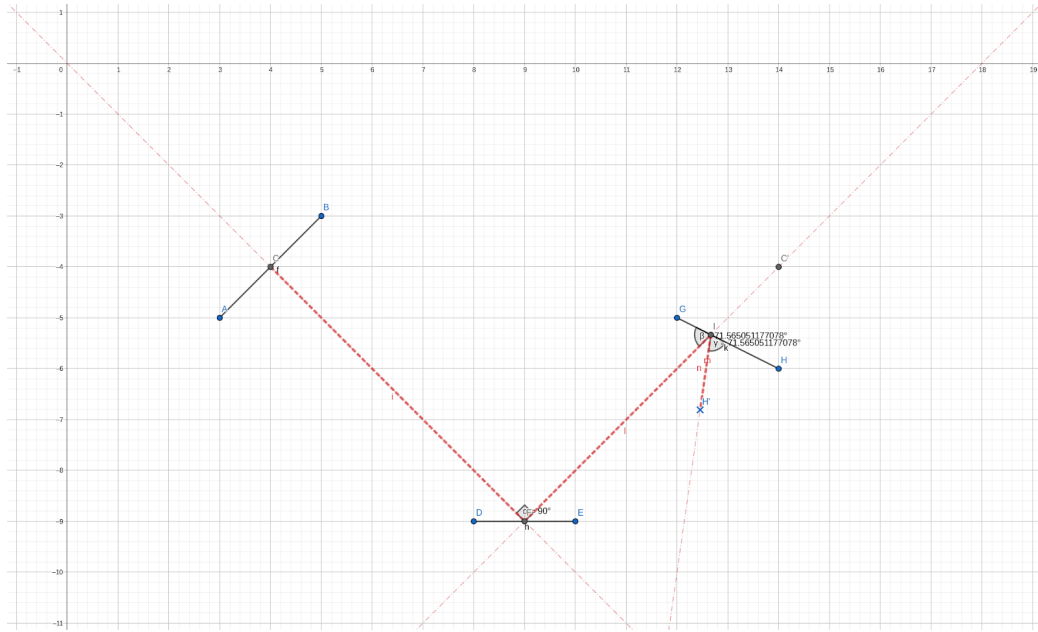


FIGURE 6 – Shémas fonctionnel du laser

4.3 Map de base

Il était conclu dans notre cahier des charges que chaque map soit composés de trois choses :

- Une fenêtre avec des murs autour pour empêcher le personnage de sortir.
- Des emplacements de miroirs afin de faire refléter le laser dessus.
- D'autres murs plus "concrets" et visibles par le joueur qui servent d'obstacle à contourner afin de réussir le niveau.

Ces quelques aspects mis ensemble nous avons réfléchi à différents niveaux permettant à termes d'obtenir un niveau jouable.

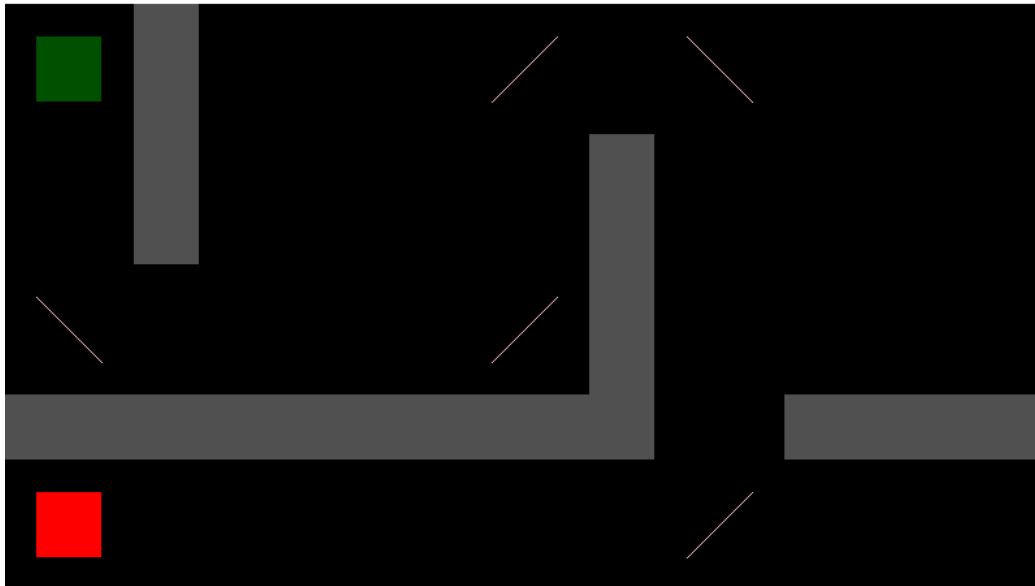


FIGURE 7 – Squelette d'un niveau

Ainsi ce niveau a été réalisé de sorte à avoir une vision d'ensemble sur les choses importantes : le carré vert correspond à la zone de départ là où le robot apparaît, les trait roses sont les emplacements des miroirs (ici d'ores et déjà dans le bon sens pour la compréhension) et les obstacles représentés en gris.

Tout ceci n'étant qu'une ébauche d'un niveau, toutes les textures sont dites "non-obstruantes" car le personnage peut les traverser, ce problème sera résolu dans un second temps.

4.4 Menu

Très succinct mais pour autant développé, ce menu consiste en trois cases rudimentaires mais néanmoins nécessaires qui sont : “recommencer” permettant de lancer une nouvelle partie, et “quitter le jeu” qui porte très bien son nom. Pour détecter l’appuie sur les cases il est nécessaire de vérifier que les coordonnées de la souris lorsque celle-ci clique soient comprises entre les limites du rectangle constituant le bouton.

Enfin, il ne reste plus qu’à appliquer les textures correspondant aux fonctions du rectangle.

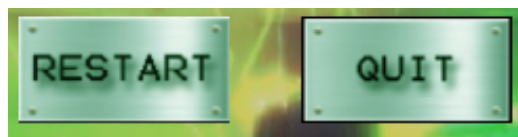


FIGURE 8 – Boutons du menu

4.5 Ajout des textures

Pour finaliser l’aspect visuel du jeu, nous avons voulu y ajouter des sprites pour habiller les murs, le sol, les objets avec lesquels on interagit. Les textures ci-dessous ont été prévues dans le but d’être implémentées pour les murs.

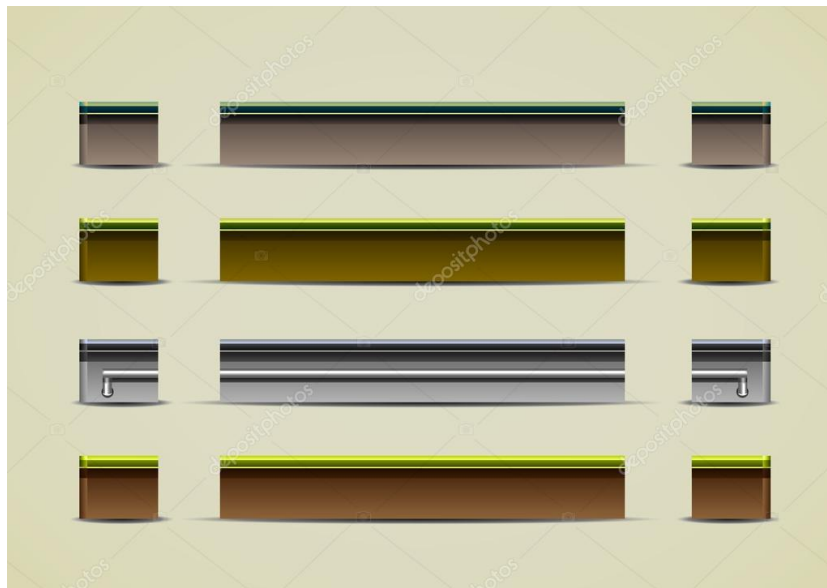


FIGURE 9 – Textures

5 Conclusion

5.1 Limitations

Il aura été clair que des limitations nous auront été imposées non pas par le projet en lui-même et ses indications, mais par l’environnement de travail. En effet il a très vite été fait au fil des séances une observation grandissante : ”Comment se mettre dans un environnement de travail à trois, à distance, sans se voir ni mise au point possible”.

Nous sommes passé par des étapes de délégations en fonction de nos connaissances et appréciations de la tâche mais est venu un autre problème, nous ne connaissions pas le langage de SDL2 et étant un langage dépassé mais néanmoins différent que sa première version de SDL, il se retrouvait un peu seul et délaissé des différents tutoriels, il a donc été compliqué de se documenter.

5.2 Améliorations

Parmi les possibilités d’améliorations s’offrant à nous, il est envisageable un rajout de la compatibilité de jeu avec une manette, pour ceci il suffira d’associer les touches d’une manette avec les actions que l’on fait au clavier. La suite logique est de faire un menu “option” permettant de changer ses contrôles, cela nécessiterait de changer les variables liées au contrôle, tout en s’assurant de ne pas avoir de redondance au niveau des actions associées aux touches, afin d’éviter un cas où la flèche de droite ne soit associé à “aller à droite” et “agrandir le jeu”. Niveau esthétique, nous pouvons imaginer choisir parmi plusieurs modèles de robots afin de personnifier le joueur à sa guise.

Enfin, conceptualiser nos idées inachevées comme un éditeur de niveau où à la place de contrôler le robot le joueur contrôle des blocs qu’il peut disposer et seront mis en mémoire pour une partie ultérieure. De plus, un scoreboard de chaque tentative où sera inscrit le temps mis à compléter le niveau ou le nombre de miroirs pivotés, tout ceci peut être mis en place grâce à un timer et une valeur incrémentée mise en sauvegarde dans un fichier texte.

5.3 Bilan

Ce projet nous aura permis en plus de nous rassembler en ce temps troublés, ainsi que les connaissances rudimentaires d’un travail en groupe et tout ce qui en découle, tel que : le partage des tâches, l’écoute et la collaboration.

De plus ce projet permet de mettre en application nos cours vus dans l’année en algorithmique, base de données et sécurité informatique, dans le

but de les adapter à un projet concret et d'en tirer parti.

Il est également important de découvrir et mettre en place des outils professionnels propices à la réalisation d'un projet, comme discord pour la communication.

D'un point de vue concret, ce type de projet permet à tout à chacun de découvrir ses limites et de se développer, ce qui aura été notre cas et nous sommes dès à présent conscients de nos capacités et ainsi plus que jamais apte à les exploiter au mieux.

6 Annexe

- Lazy Foo' Productions : Beginning Game Programming v2.0
- developpez.com : Installer SDL sous linux en ligne de commande
- devopssec : Les bases fondamentales de l'affichage graphique en SDL2
- FormationVidéo : Langage C 23 - introduction SDL