

RAPPORT DE DÉVELOPPEMENT DE CONCEPTION LOGICIEL

GUILLOTE SAMUEL
KARKASHADZE LUKA

MAHIER AWEN
LEFLAMBE Joris
TRANSON ALEXANDRE

Étudiant Université de Caen

L1 info,
Groupe 2A,

Thème: Développement d'un Puzzle Quest,

25 avril 2022



**UNIVERSITÉ
CAEN
NORMANDIE**

Sommaire

1 Présentation du projet	3
1.1 Les consignes du projet	3
1.2 Le choix de notre thème	3
2 Le SVN	3
2.1 L'utilisation du SVN durant notre projet	3
2.2 L'apport du SVN dans le projet	4
3 La Programmation Orienté Objet	4
3.1 Le principe de ce codage	4
3.2 L'apport de ce codage dans notre projet	4
4 L'élaboration de notre jeu	5
4.1 Premiers plans du jeu avec ses premières caractéristiques	5
4.2 Nos objectifs	6
4.3 L'élaboration technique de notre projet	6
4.4 Notre organisation initiale	6
4.5 Notre méthode de travail initial	6
5 La familiarisation avec notre environnement de travail	7
5.1 SVN	7
5.2 La POO	7
6 Les premiers codes	7
6.1 Les premiers fichiers de class	7
6.2 Les premiers fichiers	8
6.2.1 La première matrice	8
7 Un événement au sein de notre groupe	8
7.1 Un nouveau camarade	8
7.2 Une nouvelle organisation	8
8 La suite du déroulement de la programmation	9
8.1 Luka/Joris	9
8.2 Samuel	9
8.3 Awen/Alexandre	9
8.4 Pour la suite	9
9 Les algorithmes principaux de notre projet	10
9.1 Présentation des principaux algorithmes	10
9.2 Le menu	10
9.3 La Carte	11
9.4 Le jeu principal	12
9.5 Explication plus détaillé d'un des algorithmes	13
10 Les difficultés rencontrées	15

11 Résultats final	15
11.1 Le lancement du Jeu	15
11.2 Menu	16
11.3 Carte	16
11.4 La barre d'expérience	17
11.5 Le jeux	17
11.6 Organisation des fichiers	17
11.7 Bilan du jeu final	18
12 Mode d'emploi du jeu	19
12.1 Le lancement du jeu	19
12.2 Les règles de déplacements	19
12.3 Les combats avec l'ennemi	19
13 Bilan général du projet	20
13.1 L'apport de ce projet	20
13.2 Conclusion	20

1 Présentation du projet

Avant tout, nous allons commencer par faire une brève présentation de ce projet ayant eu lieu dans la matière "Conception Logiciel". Ce dernier ayant pour but de nous apprendre à élaborer et concevoir un logiciel, ici l'objectif est de créer un jeu vidéo.

1.1 Les consignes du projet

Le projet est contraint par un certain nombre de consignes, avec notamment l'obligation de l'utilisation de certains outils comme le SVN ou bien même la Programmation Orientée Objet que nous allons appeler POO. C'est exclusivement avec cette dernière que le projet devra être conçu. En plus de ces contraintes de conception nous avons des choix de sujet imposés parmi eux nous avons :

- Le War Games
- La simulation d'écosystèmes
- Labyrinthe 3D
- Puzzle Quest
- Block Puzzle multijoueur
- Sokoban
- Interfaces pour jeux de cartes

1.2 Le choix de notre thème

Parmi ces sujet nous avons fait le choix du Puzzle Quest. Nous l'avons choisi car c'est celui qui nous paraissait le plus complet, avec plusieurs points compliqués que nous allons devoir gérer. Avec la conception du puzzle mais des algorithmes pour permettre de jouer avec ce puzzle. Mais également avec son affichage ainsi que celui de tous ses éléments qui permettront ainsi à l'utilisateur de jouer. Nous avons aussi penser à la complexité de réalisation des différents menus nécessaires dans le jeu.

2 Le SVN

Pour la réalisation de ce projet nous avons eu la contrainte de travailler ensemble au sein d'un projet mais chacun avec sa machine et ses fichiers. C'est pour cela que nous avons utiliser le partage de fichier par Subversion (SVN) nous permettant ainsi la gestion de l'ensemble de notre projet.

2.1 L'utilisation du SVN durant notre projet

L'utilisation du SVN nous a permis de partager nos fichiers tous en gardant à jour ceux des autres membres de ce projet. Cette utilisation se faisant par le biais de commande sur le terminal nous a d'abord causé quelques problèmes, notamment au début afin de partager nos fichiers et accéder à notre projet. Après une prise en main assez compliqué avec des réflexes à avoir, le SVN s'est montré indispensable par la suite nous facilitant grandement l'avancé du projet. Avec notamment la fonctionnalité "update" qui met à jour les fichiers du projet du poste sur lequel on travaille permettant ainsi de tester très rapidement les programmes des autres avec le notre et de potentiellement repérer quelques bugs.

2.2 L'apport du SVN dans le projet

Nous le verrons tout au long de ce rapport mais le SVN à était le centre de tout notre projet. Avec la possibilité de communiquer avec les autres, mais aussi de pouvoir travailler de n'importe où. Ce qui a facilité l'avancé du projet car il n'y avait aucune contrainte de présence à un endroit précis pour coder. Il nous était donc possible de mettre en place une idée de code qui nous serait parvenu à l'esprit à un moment autre que nos heures de projet en présentiel, et de la tester sur notre programme. Permettant d'avancer de manière individuelle et indépendante sur le projet tout en informant les autres membres du groupe des changements apportés et donc un état des lieux du projet.

3 La Programmation Orienté Objet

Afin de coder notre jeu nous avons utilisé la programmation orienté Objet de python de python connu sous l'abréviation POO

3.1 Le principe de ce codage

Selon la définition de jedha La programmation Orientée Objet serait un paradigme au sein de la programmation informatique. Il s'agirait d'une représentation des choses, un modèle cohérent. Autrement dit la programmation de python se centre sur la création de classe d'objet permettant ainsi une meilleure compréhension du programme. La POO permet surtout de générer des attributs/caractéristiques à ces objets que nous allons définir.

3.2 L'apport de ce codage dans notre projet

Ce type de codage permet de faciliter la mise en place de variables utilisables dans n'importe quel programme de le classer selon leur usage comme celui pour les statistiques qui facilitent la lisibilités du code et simplifie grandement sa modification. Ainsi nous avons une véritable "arborescence" dans les diverses class de variables et leurs caractéristiques. Ce qui permet à tous les membres du groupes de les comprendre de manière rapide sans avoir la nécessité de relire la majorité du code précédemment fait. Ainsi les variables sont définies pour tous les programmes du jeu, rendant leurs codes plus lisibles donc plus rapides à faire.

4 L'élaboration de notre jeu

Maintenant que nous avons vu les outils à disposition nous allons voir l'élaboration de notre jeu.

4.1 Premiers plans du jeu avec ses premières caractéristiques

Tout d'abord nous avons imaginé la manière dont allait être codé notre jeu avec pour commencer l'imagination de toutes les classes d'objets dont on allait avoir besoin pour rendre notre jeu fonctionnel. Avec ci dessous la liste de tous les éléments dont on allait avoir besoin (cette dernière à par la suite subit de nombreux changements afin d'obtenir le meilleur résultat nous y reviendrons par la suite.

```

classe d'objet :
-case(0-10)

-classe boules:(0-4)
    -épée->0
    -bouclier->1
    -magique->2
    -défense magique->3
    -soin->4

-mouvement(boules durant le jeux)

-classe stats:
    -attaque(0-100)
    -défense(0-100)
    -magie(0-100)
    -défense magique(0-100)
    -vie(0-100)

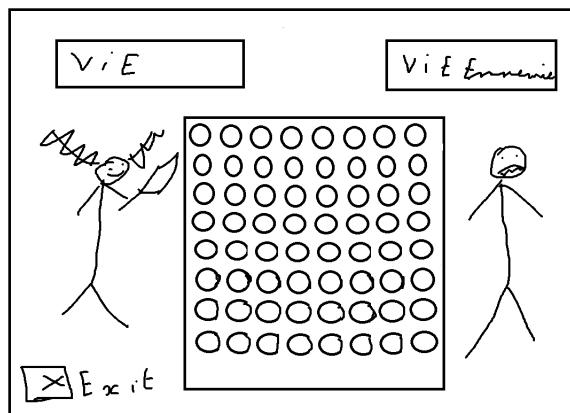
classe:
    -calcul dégat(physique-magique)
    -calcul régénération

carte:
    -localisation des niveaux(endroit où il faut cliquer pour accéder au niveau)
    -petit avatar

inventaire:
    -items

```

Suite à cela nous avons émis plusieurs croquis de conception et imagination du fonctionnement de notre jeu avec en l'occurrence le design que pouvait prendre celui-ci ainsi que son affichage (menu, puzzle, niveau, etc...) Nous avons un exemple d'un de ces croquis ci-dessous :



(Certes ce n'est pas très artistique mais ce croquis à le mérite d'exister)

Une fois que nous avions ces éléments nous avons pu imaginer la manière dont on allait le coder avec une organisation dans laquelle chacun avait ses objectifs à atteindre.

4.2 Nos objectifs

Nous nous sommes également fixés certains objectifs comme celui de créer un jeu complet avec un certain nombre de menu notamment avec un système de compétence et de d'inventaire avec des objets gagnés durant l'aventure du jeu. Nous avons aussi certain objectif comme intégrer des animations aux personnages mais aussi d'intégrer des bruitages afin de créer une ambiance au sein du jeu. Notre objectif principal reste de créer un jeu agréable à l'œil mais surtout fonctionnel avec aucun bug.

4.3 L'élaboration technique de notre projet

Pour commencer nous avons décidé de créer un fichier qui lancerai le jeu ainsi que son affichage. Ce dernier ferait appel à tous les autres fichiers et fonctions programmées dans d'autres fichiers dont chacun sera nommée par ce qu'il contient ou de sa fonctionnalité au sein du projet permettant ainsi une meilleure lisibilité dans l'arborescence de notre projet pour tous les membres de notre groupe.

4.4 Notre organisation initiale

Pour cela nous nous sommes réparti la tâche en 4 grands éléments, chacun attribué à une personne de notre groupe.

Samuel a pour objectif d'imaginer et de faire fonctionner la carte du monde du jeu avec l'animation du héro qui se déplace dans cette dernière.

Awen quant à lui a l'objectif de faire fonctionner le jeu et de créer le "jeu". Autrement réunir les différentes parties du code dont les caractéristiques et règles du jeu afin de créer une harmonie qui permettra le bon fonctionnement du jeu lors de son utilisation.

Luka a pour objectif de "fabriquer" le puzzle avec notamment la création de la grille sur laquelle reposera les niveaux du jeu. Suite à cela il s'occupera de l'écran d'accueil lors du lancement du jeu.

Alexandre doit imaginer et coder les algorithmes du jeu. Notamment pour vérifier que trois boules de même famille ne soit pas aligner dès le lancement du niveau. Mais aussi pour vérifier que tous déplacements réalisé par le joueur entraînera bien une combinaison d'au moins 3 boules de même famille.

4.5 Notre méthode de travail initial

Chaque membre du groupe ayant ses objectifs fixés, nous avons donc décidé de travailler dans un premier temps "chacun dans notre coin" afin d'avancer au plus vite dans le début de développement. Suite à cela nous réunissions à chaque fin de séance nos fichiers avec les programmes sauvegardés, dans le but de pouvoir faire les jonctions avec le fichier principal de jeu et donc de savoir ce qui marchait et au contraire ce qui ne marchait pas. Ainsi nous avons pu avancer de manière optimale pour les bases du jeu. Car en effet cette méthode est bien mais possède ses failles notamment avec le partage des fichiers lorsque le jeu commence à être "gourmand" en terme de programme. Prenons l'exemple du principe des lignes de boules qui doivent se supprimer si elles sont 3 à être alignées et seuls les déplacements entraînant cette vérification sont pris en compte. Pour cela nous avions Luka et Alexandre qui s'occupaient de faire les vérifications afin de vérifier si 3 boules étaient alignées. Tandis qu'Awen de son côté s'occupe de gérer la suppression des boules si la vérification précédemment évoquées était réalisée, il s'occupe également de l'affichage des déplacements des boules. Cette partie du développement a rendu impossible le bon fonctionnement de notre organisation précédemment évoqué car les fichiers de programmes étaient totalement dépendants les uns des autres donc

nous nous sommes entendus en fonctionnant par étape par étape. C'est-à-dire qu'une fois un des membres du groupe était sûr du bon fonctionnement de sa partie de code, il faisait la jonction avec le programme principal et partageait tout son travail aux autres membres via SVN. Ainsi les autres membres n'avaient plus qu'à mettre à jour leur SVN et donc d'accéder directement à la nouvelle étape du programme et donc de vérifier en temps réel (ou presque) du développement le bon fonctionnement de son programme et éventuellement déceler de potentiel problème/bug avec les programmes de chacun. C'est ainsi que nous avons procédé jusqu'à la fin du codage de notre projet. Cependant, nous avons eu un événement qui est venu modifier notre organisation notamment en terme de développement. Nous le verrons un peu plus loin.

5 La familiarisation avec notre environnement de travail

Avant de nous lancer dans la programmation de notre projet nous nous sommes concentré en priorité sur la découverte/familiarisation de l'environnement de travail

5.1 SVN

Nous avons d'abord eu l'obligation de le créer sur La forge Unicaen. Ainsi nous avons pu et dû apprendre les bases de l'outil SVN dont on avait parlé précédemment. Les débuts ont été fastidieux notamment pour permettre à chaque membre du groupe d'accéder au projet. Nous avons en effet mis un peu de temps afin de trouver la commande (pourtant bien exposé) "svn checkout https://forge.info.unicaen.fr/svn/group" qui permet de créer le premier dossier du projet celui qui permettra à chaque membre du groupe d'accéder aux fichiers, mais aussi de partager les leurs et de modifier ceux déjà présents.

Suite à cela nous avons pu tester les commandes pour partager puis mettre à jour le dossier sur le compte Unicaen de chaque membre.

5.2 La POO

Bien que souvent utilisée la Programmation Orientée Objet n'était pas connu de tous. Bien que certains membres du groupe ayant déjà côtoyer cette forme de programmation, d'autres au contraire n'en avait pas la moindre connaissance. Nous avons donc décidé de consacrer un moment avant de démarrer le projet durant lequel les membres qui connaissaient la POO expliquaient de manière à éclaircir les zones d'ombres que les autres membres pouvaient avoir, de sorte à être sur un pied d'égalité.

6 Les premiers codes

Maintenant que la familiarisation a été faite nous avons pu nous lancer dans la programmation de notre projet.

6.1 Les premiers fichiers de class

Les premiers fichiers codés ont été les fichiers de "class" contenant toutes les variables ainsi que toutes leurs caractéristiques. Qui par la suite, nous le verrons, seront amenés à être modifiés voire pour certaines supprimés. Nous avons commencer par créer les fichiers "class" contenant les caractéristiques du "Héros", de la "Carte" ainsi que celles des "Boules" qui seront disposées dans la matrices dont nous allons parler prochainement. Lors de l'avancé du projet

nous pourrons constaté que le fichier contenant la "class" "Boules" sera supprimé car totalement inutile.

6.2 Les premiers fichiers

Suite au codage de ces premiers fichiers nous avons pu démarrer la programmation des fichiers qui feront les bases de notre jeu ainsi que de toutes ses fonctionnalités. Nous avons tous d'abord focalisé notre attention sur le fichier de "jeu.py" dans lequel nous avons fait en sorte que le class soient utilisées dans le but de créer le puzzle dans lequel l'utilisateur sera amené lors qu'il jouera.

6.2.1 La première matrice

Le programme qui effectuent l'échanges des différentes boules était a l'origine baser sur une classe boules qui contenait sa position dans une grille et sa valeur entre (0 et 5).

On avait donc un affichage basé sur une classe sans matrice mathématique derrière, ce qui posera des problèmes pour générer la grille contenant les différentes boules. Malgré cela, cette classe permettait quand même de faire des échanges entre les boules avec une grande facilité.

7 Un événement au sein de notre groupe

Comme évoqué précédemment notre groupe à connu un événement qui a nécessité une réorganisation totale de notre méthode de travail.

7.1 Un nouveau camarade

En effet, suite à la décision de notre enseignant visant la dissolution d'un groupe contenant peu de personnes présentes et investies dans cette unité d'enseignement. Suite à cette décision, Joris, qui deviendra notre futur camarade, se retrouvé malgré lui seul. Afin de remédier à cela, notre enseignant nous a demandé si notre groupe "voulait bien accueillir un nouveau membre ? ". Tous les membres du groupe ont répondu positivement à cette requête. Ainsi le groupe s'est agrandi ce qui a entraîné deux nouveaux objectifs : -Le premier était d'intégrer le nouveau membre du groupe à notre projet et de lui expliquer toutes nos démarches de développement, mais aussi le fonctionnement de notre programme, mais aussi de l'aider dans l'apprentissage de la POO qui lui paraissait un peu abstraite et complexe. -Le second objectif était de réorganiser le fonctionnement de travail au sein du groupe afin que notre nouveau membre ne s'ennuie pas et surtout partage de potentielles idées que ce soit au sein de notre jeu ou bien dans son fonctionnement avec des optimisations de programme.

7.2 Une nouvelle organisation

Tout d'abord dès l'arrivée de Joris nous lui avons fait une brève présentation du code, sa manière de fonctionner avec ses fichiers contenant les class que nous joignons ensuite aux autres fichiers. Nous n'avons pas forcément pu expliquer plus en détail par manque de temps, mais avons tout de même laissé à Joris l'accès aux programmes, et nous sommes resté à son entière disposition pour tout problème de compréhension ou flou sur notre projet. Ainsi durant les vacances de février Joris à pu entièrement faire son intégration au sein de notre groupe et de comprendre le code nous permettant ainsi de poursuivre le développement de notre projet avec notre nouveau membre arrivé. Suite à l'intégration de Joris, nous avons donc imaginé de nouveaux objectifs pour chacun et nous avons partagé différemment les diverses tâches. Cette

arrivée a entraîné l'imagination de nouvelles mécaniques dans notre jeu et ainsi le rendre un peu plus complet avec notamment l'apport d'une sauvegarde automatique mais aussi l'apparition du menu "option" dans le menu principal du jeu, permettant justement de gérer la sauvegarde et donc de la réinitialiser, nous reviendrons sur la manière dont nous avons codé cela un peu plus tard. Ensuite nous avons redistribué les tâches aux divers membres du groupe avec un groupe se focalisant sur le jeu et sa programmation, et le second groupe se focalisant sur l'aspect graphique et le bon affichage du jeu. Le premier groupe est composé de Awen, Luka et Alexandre et le second est quant à lui composé de Samuel et de Joris. Par la suite nous verrons que ces groupes ainsi que leurs objectifs seront amenés à évoluer.

8 La suite du déroulement de la programmation

Suite aux récents changements nous allons voir ce que chacun a pu faire dans son Groupe avec les objectifs qui ont été atteint et donc l'état de l'avancement du jeu.

8.1 Luka/Joris

Tout d'abord nous allons nous pencher sur les diverses fonctionnalités à cet instant du développement, commençons par le menu qui est entièrement codés avec l'accueil programmeur par Luka et Joris qui nous permet d'avoir une véritable "coupure" entre le jeu et son accueil. Permettant d'avoir accès à un onglet, celui des options dont nous parlerons un peu plus loin. Le menu permet également l'accès à la carte dont nous allons parler de suite.

8.2 Samuel

La carte quant à elle est codé par Samuel qui se concentre sur les diverses fonctionnalités avec notamment l'animation de déplacement du héros mais aussi avec son système permettant uniquement l'accès aux niveaux "débloqués" par l'utilisateur. Samuel fera également en sorte que le personnage se déplace le plus naturellement possible avec l'aide de nombreuses

8.3 Awen/Alexandre

De leur côté Awen et Alexandre avec l'aide de Luka continue dans leur programmation des algorithmes du jeu qui possède un certains nombres de contraintes. Notamment avec le fait de vérifier s'il existe encore un déplacement possible. Pour rappel un déplacement est considéré comme possible si 3 boules de la même famille sont alignées nous y reviendrons plus en détail plus loin. Les algorithme de combats entre les deux personnages. À noter que l'intelligence artificielle des ennemis a été codés par Joris. Cette dernière empêche entre autres que les ennemis se soignent lorsque qu'ils possèdent un certain nombre de point de vie.

8.4 Pour la suite

À partir de cette nouvelle organisation nous avons pu finir poursuivre notre projet et d'arriver à un certain stade. Maintenant nous allons voir les nouveaux groupes qui vont se former ainsi que les objectifs qui succéderont à ces changements de groupes. Tout d'abord nous gardons le groupe de Joris et Luka qui vont poursuivre avec un système d'expérience du personnage tout au long de l'avancée du joueur dans le jeu. Nous avons ensuite Awen qui à rejoint Samuel dans le but d'améliorer l'affichage du jeu et le rendre plus beaux mais aussi et surtout plus fluide

dans ses animations notamment dans le menu de la carte du jeu qui était assez gourmande en terme de lignes de code de part sa conception avec bien trop de conditions qui ralentissaient le jeu tout entier. Quant au dernier membre, il s'est occupé de la sauvegarde ainsi que de son implémentation dans le jeu, nous reviendrons sur son fonctionnement un peu plus tard.

9 Les algorithmes principaux de notre projet

Maintenant que nous avons vu la manière dont on s'est organisé nous allons voir les principaux programme sur lesquelles se repose notre projet.

9.1 Présentation des principaux algorithmes

Nous allons voir maintenant les principaux programmes sur lesquels se repose notre projet.

9.2 Le menu

Tout d'abord nous avons lors du lancement du jeu le programme menu. Ce programme nous permet d'avoir des zones cliquables qui emmèneront soit vers l'onglet "option" ou bien l'onglet "carte" dont nous allons parler prochainement. De plus ce programme est celui qui va charger la sauvegarde du joueur par le biais du fichier "sauvegarde.txt". Voici l'extrait du programme :

```
perso=chargerhero(perso1, 'sauvegarde.txt')
perso.attaque=float(perso.attaque)
perso.defMax=float(perso.defMax)
perso.defense=float(perso.defense)
perso.magie=float(perso.magie)
perso.defMagieMax=float(perso.defMagieMax)
perso.defMagie=float(perso.defMagie)
perso.vie=float(perso.vie)
perso.vieMax=int(perso.vieMax)
perso.niveau=int(perso.niveau)
perso.experience=float(perso.experience)
perso.experienceMax=float(perso.experienceMax)
perso.carte_max=int(perso.carte_max)|
```

Ici nous ouvrons en lecture seul le fichier de la sauvegarde puis nous stockons toutes ses valeurs en format "string" dans les caractéristiques de la class "hero". Nous les convertissons en entier afin qu'elles soient exploitables par le programme, pour se faire nous avons utilisé la propriété "int()". Maintenant que nous avons le menu nous allons voir le programme qui permet le lancement de la carte du jeu.

9.3 La Carte

Nous avons maintenant un des programmes fondamental de notre jeu celui de la carte qui est composé de deux principales parties dans une boucles infinie. Nous avons une partie consacrée à l'initialisation du monde avec les initialisations :

- du fond de la carte

```
#initialisation de la carte#
map_monde = carte(i) # je crée la carte
screen = map_monde.screen # je change l'écran pour qu'il corresponde a l'écran de ma carte
longueur_ecran = screen.get_width() #je prend la longueur et le largeur de l'écran
hauteur_ecran = screen.get_height()

#je crée des images que je met dans des variable grace a une boucle
tab=[1,2,3,4,5]
for i in tab:
    globals()["monde"+str(i)+"_jouer"] = pygame.transform.scale(pygame.image.load("image_carte/carte_jeu_jouer_"+str(i)+".png"),(longueur_ecran,hauteur_ecran))
for i in tab:
    globals()["monde"+str(i)] = pygame.transform.scale(pygame.image.load("image_carte/carte_jeu_jouer_"+str(i)+".png"),(longueur_ecran,hauteur_ecran))
```

Ici nous avons le fond de la carte qui est l'arrière plan sur lequel le héros va se déplacer dans le menu

```
#change le fond
fond = pygame.image.load("image_carte/carte_jeu_jouer_1.png")
map_monde.screen.blit(map_monde.fond,(0,0))
map_monde.screen.blit(map_monde.hero,(37*map_monde.longueur_ecran/
1280,110*map_monde.hauteur_ecran/720))
pygame.display.flip()
```

- du héros (qui sera utilisé durant la partie du joueur)
- des zones "cliquables" qui permettent aux joueurs de cliquer sur un niveau afin d'y accéder.

```
#initialisation des zones clickable en utilisant la même méthode que précédament avec des
tuples
tab_xy = [(440*map_monde.longueur_ecran/1280,628*map_monde.hauteur_ecran/720),-
(757*map_monde.longueur_ecran/1280,631*map_monde.hauteur_ecran/720),(53*map_monde.longueur_ecran/
1280,111*map_monde.hauteur_ecran/720),(194*map_monde.longueur_ecran/
1280,473*map_monde.hauteur_ecran/720),(575*map_monde.longueur_ecran/
1280,259*map_monde.hauteur_ecran/720),(991*map_monde.longueur_ecran/
1280,557*map_monde.hauteur_ecran/720),(1086*map_monde.longueur_ecran/
1280,62*map_monde.hauteur_ecran/720)]
tab_taille = [(300*map_monde.longueur_ecran/1280,150*map_monde.hauteur_ecran/720),-
(40*map_monde.longueur_ecran/1280,40*map_monde.hauteur_ecran/720),(150*map_monde.longueur_ecran/
1280,150*map_monde.hauteur_ecran/720),(150*map_monde.longueur_ecran/
1280,150*map_monde.hauteur_ecran/720),(150*map_monde.longueur_ecran/
1280,150*map_monde.hauteur_ecran/720),(150*map_monde.longueur_ecran/
1280,150*map_monde.hauteur_ecran/720),(150*map_monde.longueur_ecran/
1280,150*map_monde.hauteur_ecran/720)]
for i in range(len(tab_xy)):
    globals()["zone_clickable "+str(i)]= pygame.Rect((tab_xy[i][0],tab_xy[i][1]),-
(tab_taille[i][0],tab_taille[i][1]))
```

Ici nous avons définis les coordonnées des zones cliquables et nous les avons stockées dans des variables pour rendre plus facile leur accès

- de la boucle qui sera la base de notre programme

Nous avons maintenant la seconde partie de cet algorithme qui est nécessaire au bon fonctionnement du jeu. Nous avons un grand nombre de fonctionnalités mais nous allons voir la plus importante de cette boucle qui est la vérification des zones possibles à atteindre par le joueur avec ce programme suivant : qui vérifie pour une zone

Cette partie de programme permet de vérifier si une zone est atteignable si elle l'est alors le héros s'y déplace. Ce programme est réutilisé à 4 autres reprises pour vérifier les autres niveaux.

```
#initialisation des listes pour la boucle
tab_ratio = [(37*map_monde.longueur_ecran/1280,110*map_monde.hauteur_ecran/720),-
(177*map_monde.longueur_ecran/1280,475*map_monde.hauteur_ecran/720),(558*map_monde.longueur_ecran/
1280,260*map_monde.hauteur_ecran/720),(970*map_monde.longueur_ecran/
1280,556*map_monde.hauteur_ecran/720),(1070*map_monde.longueur_ecran/
1280,64*map_monde.hauteur_ecran/720)] #zone_clickable_0
```

Ces variables seront utilisées tout au long du programme et seront amenées à changer pendant l'exécution de la boucle.

```
#permet d'aller a la zone 1#
if event.button == 1:
    if zone_clickable_2.collidepoint(event.pos) :
        zone_depart = map_monde.zone
        zone_depart_temp = zone_depart
        zone_arrive = 1
        if zone_depart > zone_arrive :
            while zone_depart_temp != zone_arrive:
                trajet = "map_monde.sprite_zone"+str(zone_depart_temp)-
                +"_zone"+str(zone_depart_temp-1)+"()"
                exec(trajet)
                zone_depart_temp = zone_depart_temp-1
        map_monde.fond = monde1_jouer
        map_monde.screen.blit(map_monde.fond,(0,0))
        map_monde.screen.blit(map_monde.hero,(tab_ratio[map_monde.zone-1]
        [0],tab_ratio[map_monde.zone-1][1]))
        pygame.event.clear()
        pygame.display.flip()
```

9.4 Le jeu principal

Avant tout nous allons expliquer la manière dont nous avons procédé afin de créer le jeu. Pour ce faire nous avons créé deux matrices une première qui est celle de "base" et qui est visible par le joueur. Puis nous en avons une seconde qui est la matrice de "contrôle" construite de deux matrices. Elles recensent toutes les interactions possibles entre les différentes boules. L'une d'elles vérifie l'alignement de 3 boules dans la première matrice en colonne. L'autre à la même fonctionnalité mais cette fois en ligne.

```
def horizontal(matrice):#crée une matrice composée de 0 et de 1 avec sachant que 0 signifie qu'il y a
des boules aligné en horizontal
    Boule_Testee=matrice[0][0]
    coordonnee_Boule_Testee=[0,0]
    liste_effet=[]
    matrice_test=[[1 for i in range (len(matrice[0]))]for j in range (len(matrice))">#crée une
matrice pleine de 1
    for i in range(len(matrice)):
        Nb_Boules_Identiques=0
        for j in range(len(matrice[i])):
            if Boule_Testee==matrice[i][j]:
                Nb_Boules_Identiques=Nb_Boules_Identiques+1
                if j==len(matrice[i])-1:#vérifie si malgré le saut de ligne il y a un
alignement
                    if Nb_Boules_Identiques>=3:
                        liste_effet.append((Boule_Testee,Nb_Boules_Identiques))#ajoute a liste effet Boule_Testee et
Nb_Boules_Identiques qui vont permettre de faire des dégâts,de soigner,de se protéger.
                    for k in range(Nb_Boules_Identiques):
                        matrice_test[i][j-k]=0
                else:
                    if Nb_Boules_Identiques>=3:
                        liste_effet.append((Boule_Testee,Nb_Boules_Identiques))#ajoute
a liste effet Boule_Testee et Nb_Boules_Identiques qui vont permettre de faire des dégâts,de
soigner,de se protéger.
                    for k in range(Nb_Boules_Identiques):
                        matrice_test[i][j-k-1]=0
                Nb_Boules_Identiques=1
                Boule_Testee=matrice[i][j]
                coordonnee_Boule_Testee=[i,j]
    return(matrice_test,liste_effet)
```

Cette partie de programme permet de créer une matrice contenant uniquement des 1. Ces derniers peuvent être remplacés par des 0 qui représentent les combinaisons possibles. Nous avons le même type de programme afin de vérifier les combinaisons verticales.

Maintenant nous allons voir "l'assemblage" de des deux matrices par le biais de la fonction ci-dessous :

```
def combinaison_de_matrice(matrice):#crée une combinaison entre la matrice vertical et horizontal et aussi des liste
    matrice_test_horizontal=horizontal(matrice)
    matrice_test_vertical=vertical(matrice)
    liste_effet=matrice_test_horizontal[1]+matrice_test_vertical[1]#crée la liste d'effet complète
    for i in range(len(matrice)):
        for j in range(len(matrice[i])):
            matrice[i][j]=matrice[i][j]*matrice_test_horizontal[0][i][j]*matrice_test_vertical[0][i][j]
    return(matrice,liste_effet)
```

Cette partie du programme permet de mettre en commun les deux programmes précédemment évoqués et donc de créer une matrice qui renvoie les combinaisons de toutes les matrices du programme permettant de savoir où les combinaisons sont possibles. Si elles sont possibles alors les boules se cassent et sont remplacées par des 0 dans la matrice. Ce qui correspond à du vide alors nous avons imaginée une fonction qui ajoute de nouvelles boules en les faisant tomber, dont voici le code :

```
while 0 in np.array(matrice):#boucle tant que il y a des zéros dans la matrice
    for j in range(len(matrice)):
        for i in range(len(matrice[j])):
            boule(i,j,tailleBoule,x,y,matrice,screen)#exécute la fonction boules
            if matrice[i][j]==0 and i==0:#rédéfinie les boules qui sont sur la première ligne
                matrice[i][j]=randint(1,5)
            elif matrice[i][j]==0:#fait chuter les boules
                matrice[i][j]=matrice[i-1][j]
                matrice[i-1][j]=0
            pygame.display.update()
```

Cette boucle continue tant qu'il y a des 0 dans la matrice. Elle à pour but de faire tomber les boules donc en cas de 0 en dessous de sa position alors la boules descend. Si jamais un 0 est sur la première ligne alors le programme va générer aléatoirement une boule par le biais d'un "randint" qu'il placera dans la matrice.

9.5 Explications plus détaillées

Comme précédemment évoqués nous allons parler de l'algorithme permettant la vérification de la présence de match potentiel (match signifie que 3 boules de même familles sont alignées). Tout d'abord nous avons séparé le programme en deux grandes parties avec une première qui regroupe plusieurs fonctions assez longue mais factoriser en quelques lignes par Luka. Comme nous pouvons le voir ci-dessous :

```
def haut(matrice,i,j):#-----verifier les different matches possible vers le haut-----
    if matrice[i][j]==matrice[i-1][j] and matrice[i][j]==matrice[i-2][j+1] or
    matrice[i][j]==matrice[i-1][j] and matrice[i][j]==matrice[i-2][j-1] or matrice[i]-[j]==matrice[i-2][j] and matrice[i][j]==matrice[i-1][j+1] or matrice[i]-[j]==matrice[i-2][j] and matrice[i][j]==matrice[i-1][j-1] or matrice[i]-[j]==matrice[i-1][j+1] and matrice[i][j]==matrice[i-2][j+1] or matrice[i]-[j]==matrice[i-1][j-1] and matrice[i][j]==matrice[i-2][j-1]:
        return True
```

Chacune de ces fonctions vérifient les matchs possibles dans une direction différente ici se sont les matchs vers le haut qui sont vérifiés. À savoir que chacune des directions est vérifiées afin d'être certain de l'inexistence de matchs.

À la suite de cette première partie nous avons la seconde qui elle a pour but de regrouper les différentes vérifications en fonction de l'emplacement dans la matrice des boules vérifiées par l'algorithme. Prenons l'exemple de la fonction vérifiant les matchs possibles sur la hauteur tout en excluant les bords de la matrice :

```

-----VERIFIER LA HAUTEUR-----
    if y>=1 and y<=len(matrice)-2:
        if x>=0 and x<=len(matrice)-3:
            if bas(matrice,x,y):
                return True
        elif x>=2 and x<=len(matrice)-1:
            if haut(matrice,x,y):
                return True
        elif x>=0 and x<=len(matrice)-1:
            if y>=0 and y<=len(matrice)-4:
                if ligneD(matrice,x,y):
                    return True
        elif y>=3 and y<=len(matrice)-1:
            if ligneG(matrice,x,y):
                return True

```

Nous avons décidé de procédé de cette manière car les vérification à effectuer ne sont pas les même en fonction de l'emplacement des boules contenues dans la matrices. Ici nous partons sur le principe des if et elif permettant de vérifier si un match est possible en hauteur dans le cas où un match est possible l'algorithme renverra TRUE et par conséquent s'arrêtera de suite. À contrario l'algorithme passe à la vérification suivante et ce ainsi de suite. Si jamais le programme ne détecte aucun match possible alors il fait appel à la fonction "consmatrice" qui permet de recréer une nouvelle matrice avec la présence de matchs. L'algorithme est le suivant

```

def consmatrice(a,b):
    matrice=[]
    for i in range(a):
        ligne=[]
        for k in range(b):
            r=randint(1,5)
            ligne.append(r)
        matrice.append(ligne) #génération de la matrice de jeu
    e=False
    while e==False:
        e=True
        z=0
        for val in range(a):
            for val2 in range(1,b-1):
                if matrice[val][val2]==matrice[val][val2-1] and
matrice[val][val2]==matrice[val][val2+1]:#on vérifie si il n'y a pas 3 boules de
la même famille en ligne
                    al=matrice[val][val2]
                    z+=2
                    while al==matrice[val][val2]:
                        matrice[val][val2]=randint(1,5)
                for val3 in range(1,a-1):
                    for val4 in range(b):#on vérifie si il n'y a pas 3 boules
de la même famille en collone
                        if matrice[val3][val4]==matrice[val3-1][val4] and
matrice[val3][val4]==matrice[val3+1][val4]:
                            m=matrice[val3][val4]
                            z+=2
                            while m==matrice[val3][val4]:
                                matrice[val3][val4]=randint(1,5)
                if z!=0:
                    e=False
    return matrice

```

Cet algorithme se divise en deux parties la première qui se contente de créer une simple matrice composée de nombre entre 1 et 5 (qui représentent les diverses familles de boules) par le biais d'une boucle for. Puis à cette matrice on y applique une boucle while dans laquelle nous allons vérifier si il n'y a pas plus de 3 nombres alignées dans la matrice. Cette boucle while est constituée de deux boucle for qui vérifie si les nombres alignées dans les lignes de la matrice. Et l'autre boucle quant à elle vérifie ceux en colonne. En cas de match la valeur sur laquelle est l'algorithme change. Et enfin pour terminer la boucle while nous avons une vérification qui permet de savoir si aucun changement n'a été fait, signifiant donc que la matrice ne comporte aucun match. Dans les cas contraires la boucle while recommence.

10 Les difficultés rencontrées

Maintenant que nous avons évoqués les principales étapes de codage nous allons voir les principaux problèmes auxquels nous avons du faire face durant le projet, notamment avec le dysfonctionnement du fichier principal du jeu qui ne reconnaissait plus les fichiers contenant les programmes de chacun, en effet ces derniers ont été mis hors état de fonctionnement à cause d'un problème de "commit" d'un membre du groupe qui sans explication apparente aurait "commit" une version antérieure du projet ce qui a causé une duplication de certains programmes plus anciens rendant ces derniers incompatibles avec ceux qui avaient déjà étaient partager auparavant créant un problème de version des fichiers. Ce problème a été résolu par la suppression des fichiers de programmes erronés et par le partage d'une nouvelle version du projet totalement remis à jour qui enlevait toutes les incompatibilités. Ce petit désagrément aura été le plus important que l'on aura eu à gérer durant la totalité du projet. Nous aurons cependant, par la suite quelques petits problèmes à plusieurs endroits du programme à signaler mais rien de majeur ou de compromettant et qui seront minimes et très simples à corriger.

11 Résultats final

Maintenant que nous avons vu les principales étapes de développements ainsi que les caractéristiques de notre projet/jeu. Nous allons voir un bref aperçu du résultat du jeu avec l'explication de chaque menu et de chaque fonctionnalité.

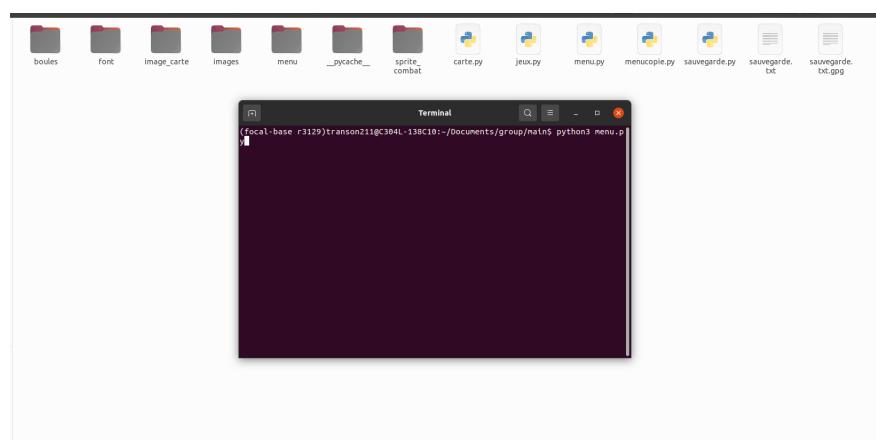
11.1 Le lancement du Jeu

Tout d'abord nous devons lancer le jeu pour se faire nous devons allons voir comment cela ce fait.

Tout d'abord nous devons aller dans le dossier "main"



Une fois arrivé ici, nous devons lancé le terminal dans lequel nous écrirons la commande d'exécution "python3 menu.py" comme ci-dessous :



Une fois cela fait nous arrivons sur le menu du jeu où il ne reste plus qu'à cliquer sur jouer pour entrer sur la carte du monde du jeu.



11.2 Menu

Une fois arrivé dans le menu (présenté ci-dessus) nous avons plusieurs choix soit de jouer comme nous l'avons précédemment évoqué soit d'accéder au menu d'option dans lequel nous avons décidé de donner la possibilité à l'utilisateur de réinitialiser sa sauvegarde et donc de recommencer le jeu (à savoir que l'utilisateur se doit de redémarrer le jeu afin que sa réinitialisation soit belle et bien effectué)

11.3 Carte

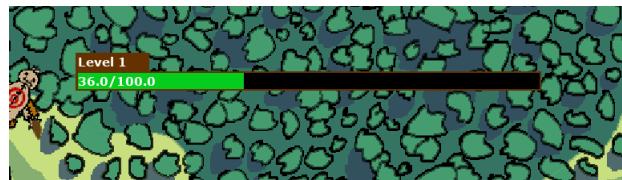
Maintenant que nous avons vu ce que le menu donne nous allons voir ce qu'affiche le programme lorsque l'utilisateur clique sur le bouton "Jouer". Voici sur quoi l'utilisateur arrive. Sur la carte le joueur ne possède pas une liberté totale. L'avancée dans le monde se mérite !



Nous avons fait le choix de bloquer le joueur au niveau auquel il est arrivé autrement dit s'il est au niveau 1 et qu'il ne l'a toujours pas accomplis il ne pourra donc pas accéder aux autres niveaux. Si le joueur à la possibilité de lancer le niveau alors il ne lui reste plus qu'à cliquer sur le bouton "JOUER" afin d'accéder au jeu.

11.4 La barre d'expérience

À noter que nous avons également intégré une barre d'expérience qui permet à l'utilisateur de consulter l'avancée de son héros notamment en niveau donc en statistiques en revanche ces dernières ne sont pas consultables.



11.5 Le jeux

Maintenant que nous avons vu toute la partie menu de notre projet nous allons le principal point c'est à dire le jeu auquel l'utilisateur va jouer. Il se présente ainsi : Nous avons imaginé



l'affichage du jeu ainsi. Avec notre héros qui est à gauche avec l'ennemi du niveau concerné ainsi que les barres de vie de chaque acteur du niveau juste au dessus d'eux. Nous avons également voulu mettre la grille de jeu ainsi afin de permettre une meilleure maniabilité pour l'utilisateur. Nous avons également imaginé une petite croix en bas à gauche qui permet de quitter le niveau en cours de jeu.

Nous avons également imaginé un affichage juste à coté de la barre de vie de chaque personnage les boules que chacun à utilisées ce qui donne ce résultat en jeu :



11.6 Organisation des fichiers

Étant donné que nous venons de voir le résultat final de notre jeu nous allons voir maintenant son arborescence. Nos programmes sont séparés en deux catégories les principaux et les

secondaires. Les principaux sont les programmes qui représentent différentes phases du jeux (menu, carte, jeux) situé dans le fichier. Les secondaires sont quant à eux des programmes qui accompagnent la bonne exécutions des programmes principaux et sont situé dans les différents dossiers a l'intérieur de main.

Les différentes images utilisées dans l'intégralité du jeux sont contenues dans des dossiers afin d'éviter la surcharge d'informations dans le dossier contenant les programmes principaux.



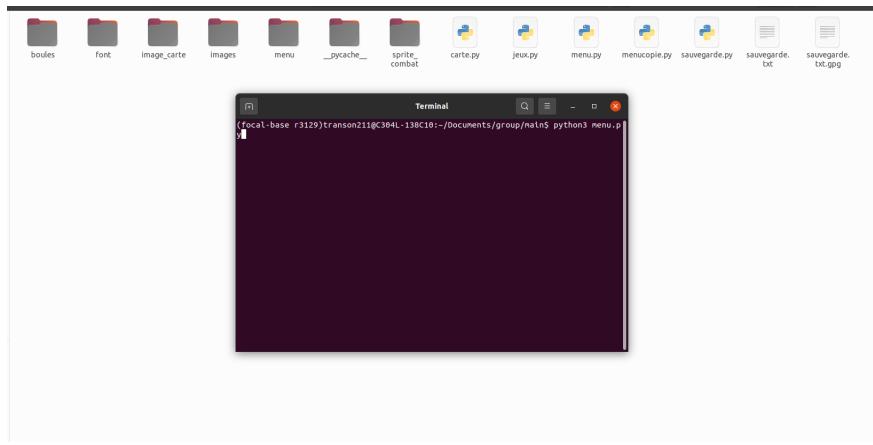
11.7 Bilan du jeu final

Nous pouvons à ce stade faire une comparaison entre le résultat final et les idées que l'on avait au début de notre projet. Tout d'abord nous sommes assez satisfait sur le plan du déroulement de ce projet en effet, nous avons su travailler en groupe et dans une bonne ambiance en général et ce malgré les problèmes rencontrés durant ce projet. Nous sommes relativement satisfaits du jeu ainsi que de son contenu et de ses graphismes, même si à notre goût il manque quelques petits détails mais nous y reviendrons. Et nous sommes surtout satisfaits sur les algorithmes de jeu avec les vérifications d'alignement d'au moins 3 boules de même famille, mais aussi de ceux qui permettent de vérifier s'il existe au moins un déplacement "valable" dans la grille de jeu, sans parler de l'algorithme de la carte qui empêche l'accès aux autres niveaux tant que le niveau précédent n'a pas été accomplis. L'ajout de la sauvegarde du jeu satisfait également le résultat final attendu voire même sur ce point dépasse l'objectif initial. Mais nous n'avons pas que des réussites dans ce projet car rien n'est parfait (et heureusement)... Tout d'abord nous regrettons un certain nombre de points sur lesquels nous aurions aimé nous attarder un peu plus. Notamment dans les menus et plus précisément ceux de la carte, en effet l'objectif initial était de créer un système d'expérience complet avec la possibilité de débloquer des compétences ... Mais de permettre au joueur d'équiper son personnage de manière complète... Par manque de temps et surtout par ordre d'importance (car ceci ne constituait pas une priorité par rapport à d'autres programmes fondamentales au fonctionnement du jeu) nous avons, par conséquent, préféré oublié ce système avec en guise de remplaçant un simple système de gain de niveau automatisés et mais qui reste consultable par le joueur avec la barre d'expérience en haut à gauche de la carte. En plus de cela, l'autre principal regret que nous avons concerne la sauvegarde... En effet nous avons eu la volonté de rendre son fichier inaccessible pour le joueur afin d'éviter toutes potentielles triches de ce dernier. Notre objectif était de crypter la sauvegarde de l'utilisateur afin de la rendre accessible uniquement par le jeu, bien que nous ayons réussi à crypter la sauvegarde, cette dernière n'était pas retranscrite correctement par le jeu la rendant corrompue. Rendant impossible le lancement du jeu. Nous avons donc été dans l'obligation d'abandonner cette idée (pour des raisons évidentes) et de préférer le bon fonctionnement du jeu à une tentative de système anti-triche bancale et inexploitable.

12 Mode d'emplois du jeu

12.1 Le lancement du jeu

Afin de lancer le jeu vous devez ouvrir le dossier "main" puis y ouvrir un terminal de commande dans lequel il vous faut écrire la commande suivante "python3 menu.py"



12.2 Les règles de déplacements

Lorsque vous avez atteint le premier niveau vous accéder à une grille de puzzle dans laquelle vous pouvez voir divers types de boules, afin de pouvoir intervertir deux boules vous devez cliquer sur la boule, maintenir le clic sur la boule puis diriger le curseur de votre souris vers la boule que vous voulez intervertir. Vous pouvez déplacer les boules de gauche à droite et de haut en bas. Afin qu'un déplacement soit considéré comme valide il faut qu'il permette de former une ligne de minimum 3 boules de la même famille.

12.3 Les combats avec l'ennemi

Durant votre partie vous devez combattre un ennemi pour vous aider vous avez à disposition différents familles de boules qui sont les suivantes :

- Les boules Jaunes avec un bouclier à l'intérieur : représentent la défense physique qui contrent les attaques physiques de l'ennemis
- Les boules Vertes avec des croix à l'intérieur : représentent le soin peuvent être apporté à votre héro afin d'augmenter sa barre de vie
- Les boules Bleues avec un spectre à l'intérieur : représentent une attaque magique que vous infligez à l'ennemi
- Les boules Rouges avec une épée à l'intérieur : représentent une attaque physique que vous infligez à l'ennemi
- Les boules Roses avec un champ magnétique à l'intérieur : représentent la défense magique qui fera face aux attaques magiques de l'ennemi

Avant tout il faut savoir que plus vous allignez de boules de la même famille plus l'effet de cette famille est conséquent.

Chaque boules ayant ses caractéristiques et impacts sur le personnages nous allons voir maintenant comment se déroule le combat durant la partie. C'est un style de combat tour par tour. Lorsque vous avez fait un déplacement valide sur le puzzle votre tour est considéré comme "joué" c'est alors au tour de l'ennemi qui s'applique l'effet d'une des cinq familles de boules. Le combat se termine lorsqu'un des deux combattants atteint un total de point de vie de 0.

13 Bilan général du projet

Nous arrivons à la fin de ce rapport de développement et donc de ce projet. Pour cela nous allons exprimer/expliquer ce que nous a apporté le projet de manière générale

13.1 L'apport de ce projet

Ce projet nous a apporté un certain nombre de choses, en effet, ce dernier nous a tout d'abord appris l'utilisation de SVN et nous a introduit à la POO. Mais ce projet nous a surtout appris toutes les étapes nécessaires à la création d'un logiciel que ce soit l'étape d'imagination/conception de ce logiciel, la réalisation de ce dernier puis enfin la présentation du rendu final. Au cours de notre projet nous avons surtout appris une chose, et non des moindres, car il s'agit du "codage papier". En effet, nous avons appris (grâce à quelques remontrances) à écrire sur feuille notre programme et de faire un certains nombre de schémas afin de concevoir la meilleure méthode de codage, mais également dans le but de ne pas nous perdre dans notre programmation. Le principal apprentissage que nous ayons reçu durant ce projet est le travail en groupe et surtout la cohésion primordial à la réalisation de ce dernier. Et enfin pour terminer ce projet nous a certes appris le travail d'équipe au sein d'un groupe physique mais il nous également appris le travail en groupe virtuel par le biais de l'environnement SVN mis à disposition durant tout le déroulement de ce projet.

13.2 Conclusion

Pour finir nous pouvons dire que durant ce projet nous avons faits face à plusieurs problèmes et appris un certain nombre de choses mais aussi à nous introduire dans un environnement de développement en groupe. Nous trouvons ce projet intéressant car c'est certainement celui qui se rapproche le plus du fonctionnement en entreprise nous permettant de nous préparer mais surtout de nous montrer la coordination d'équipe nécessaire pour la conception d'un projet professionnel. Cet enseignement nous a permis de travailler sur un projet concret et ce de manière concrète.