ECE PARIS GRADUATE SCHOOL OF ENGINEERING

Rapport de projet : SNOOPY'S REVENGE

Juliette HEUANGTHEP & Axel CANTE (TD 04)
ECE PARIS | ING2 – PROMO 2019

PRESENTATION DU SUJET

Dans le cadre de l'enseignement *C++ : Programmation orientée objet* de la 2nde année du cycle préparatoire aux études d'ingénieur, il nous a été proposé de réaliser le jeu console « La Revanche de Snoopy ».

Il s'agit d'un jeu de réflexion qui consiste à contrôler Snoopy de sorte qu'il récupère quatre oiseaux dans un temps limité tout en évitant les pièges (ex : balle rebondissante, blocs piégés, …) afin de passer au niveau suivant et augmenter son score.

Le cahier des charges nous indique qu'il faut un personnage : Snoopy, quatre oiseaux à récupérer, une balle qui tue Snoopy lorsqu'il le touche et des blocs (poussables, cassables et piégés, ces derniers tuent Snoopy).

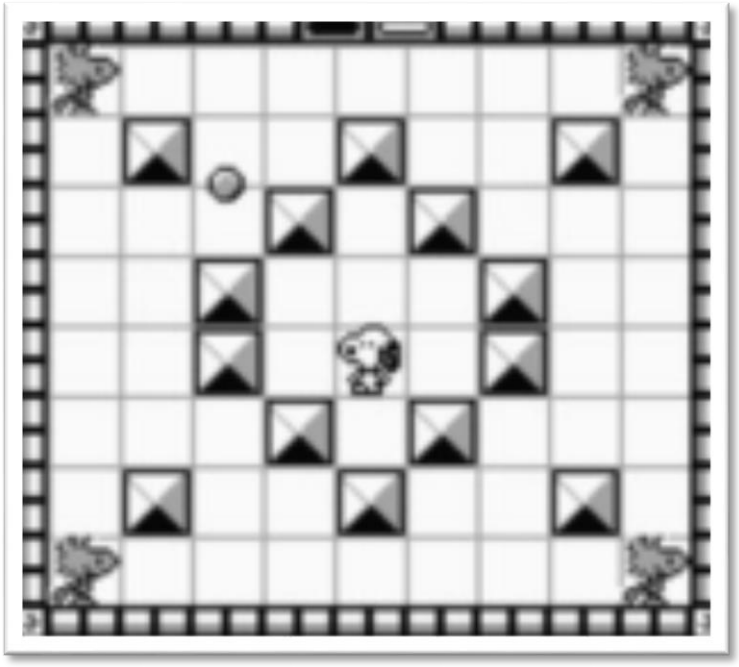


Figure 1 Snoopy's Magic Show, jeu vidéo créé en 1990

SOMMAIRE

PARTIE CONCEPTION

I. Présentation de l'équipe

Nous sommes une équipe de deux ex-Prépac : CANTE Axel et HEUANGTHEP Juliette su TD04 ING2.

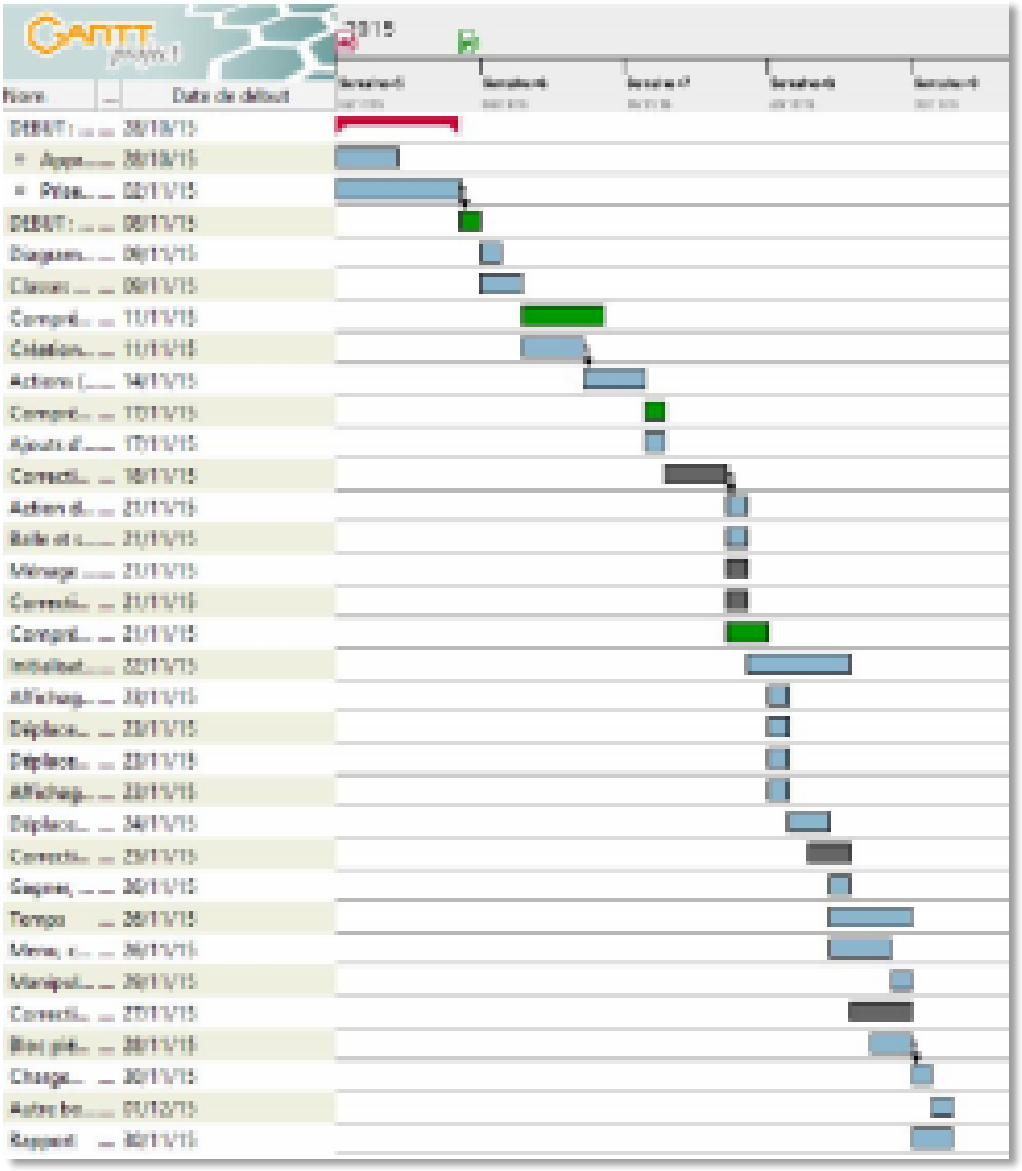
Le fait que nous soyons issus de la même classe d'ING1 accélérée a facilité notre entente même si nous ne nous connaissions pas. Ceci nous a permis d'être efficaces en termes d'organisation et de cohésion de groupe.

Nous nous sommes naturellement organisés comme suit :

Nous avons débuté par la théorie : le C++, en nous exerçant sur les TP et cours de C++ donnés par M. Diedler ainsi que OpenClassroom, et le versionning grâce aux cours de M. Cros et M. Pinto. Nous avons de même étudié le cahier des charges et établi des diagrammes de classe.

Puis, nous sommes passés à la pratique en nous entraidant lors de nos séances de travail.

Planning : organisation de l'équipe



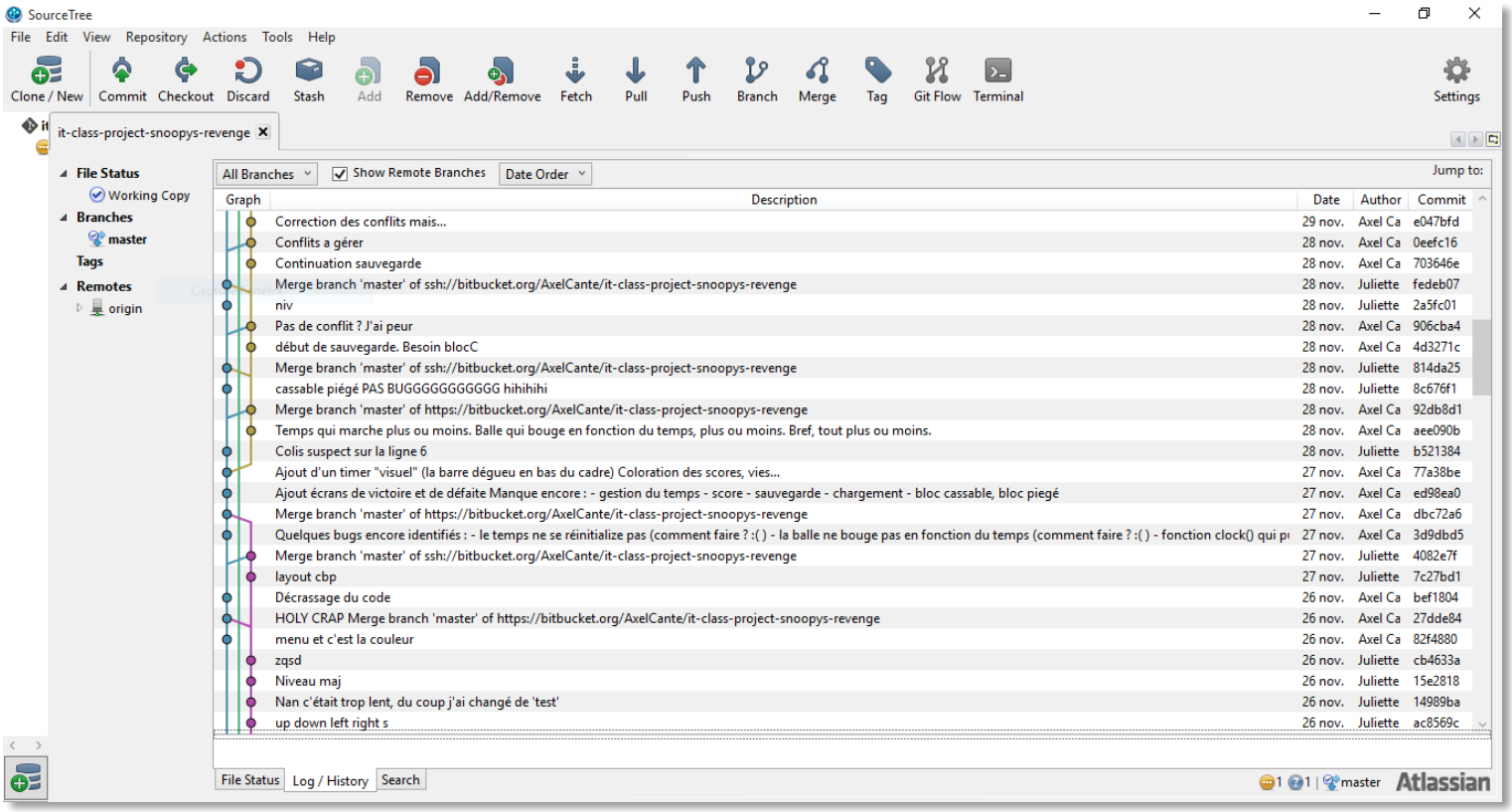
C'est le **diagramme de Gantt** (logiciel utilisé pour planifier notre projet).

Nom	Date de fin	
DEBUT : première rencontre et début de C++	07/11/15	28/10/15
Apprentissage du C++ : faire les TP1, TP2, TP3	04/11/15	28/10/15
Prise de connaissance du sujet plus en détail	07/11/15	02/11/15
DEBUT : Versioning, la naissance d'une nouvelle passion	08/11/15	08/11/15
Diagramme des classes et ACD	09/11/15	09/11/15
Classes normales, mère et filles,encapsulation	10/11/15	09/11/15
Compréhension profonde de Sourcetree	14/11/15	11/11/15
Création de la matrice de jeu	13/11/15	11/11/15
Actions (pousser, déplacer, permuter)	16/11/15	14/11/15
Compréhension de la console et de ses fonctions	17/11/15	17/11/15
Ajouts d'éléments statiques sur la matrice	17/11/15	17/11/15
Corrections de bugs	20/11/15	18/11/15
Action de pousser	21/11/15	21/11/15
Balle et son déplacement	21/11/15	21/11/15
Ménage dans le code, ajouts d'attributs, ..	21/11/15	21/11/15
Corrections de bugs, de fonctions	21/11/15	21/11/15
Compréhension plus poussée de SourceTree	22/11/15	21/11/15
Initialisations matrice et éléments	26/11/15	22/11/15
Affichage de matrice	23/11/15	23/11/15
Déplacement Balle avec un sleep	23/11/15	23/11/15
Déplacement Snoopy	23/11/15	23/11/15
Affichage de cadre plus joli	23/11/15	23/11/15
Déplacement de la balle optimisé	25/11/15	24/11/15
Correction de bugs et ménage	26/11/15	25/11/15
Gagner, perdre, mourir, augmenter le niveau, score, niveau	26/11/15	26/11/15
Temps	29/11/15	26/11/15
Menu, couleur, autres affichages comme le temps, victoire, défaite	28/11/15	26/11/15
Manipulation de la console	29/11/15	29/11/15
Correction de bugs et ménage	29/11/15	27/11/15
Bloc piégé, sauvegarde, bloc cassable,	29/11/15	28/11/15
Chargement	30/11/15	30/11/15
Autre bonus	01/12/15	01/12/15
Rapport	01/12/15	30/11/15

Nous avons utilisé GanttProject afin de réaliser un planning de ce qui a été établi durant la mise en place du projet.

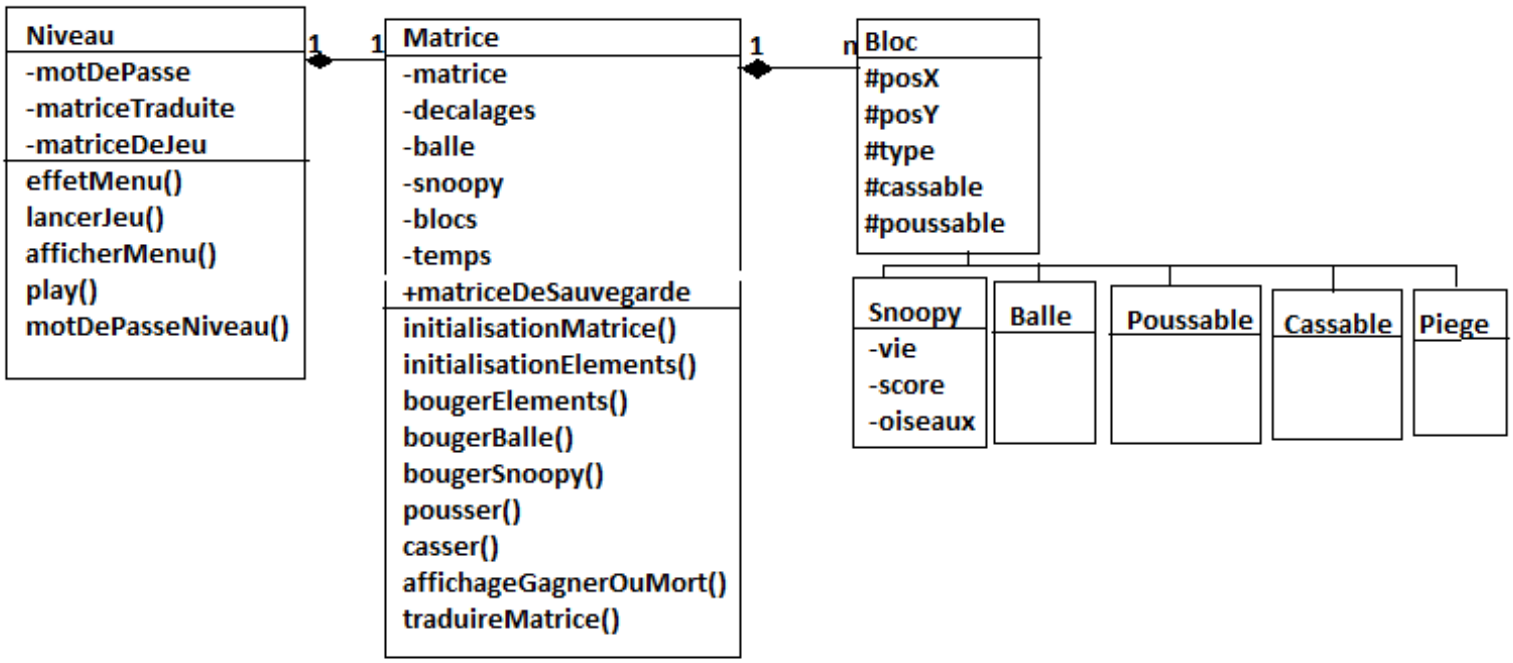
Pour plus de précision, le code couleur est le suivant :

- rouge : les étapes importantes de début ;
- bleu : l'avancement dans le code ;
- vert : la compréhension approfondie de SourceTree (* voir page suiante);
- gris : ménage et correction de bugs.



Ceci est le logiciel SourceTree qui permet d'utiliser gitbucket afin de versionner le projet en équipe.

I. Diagramme de classes général



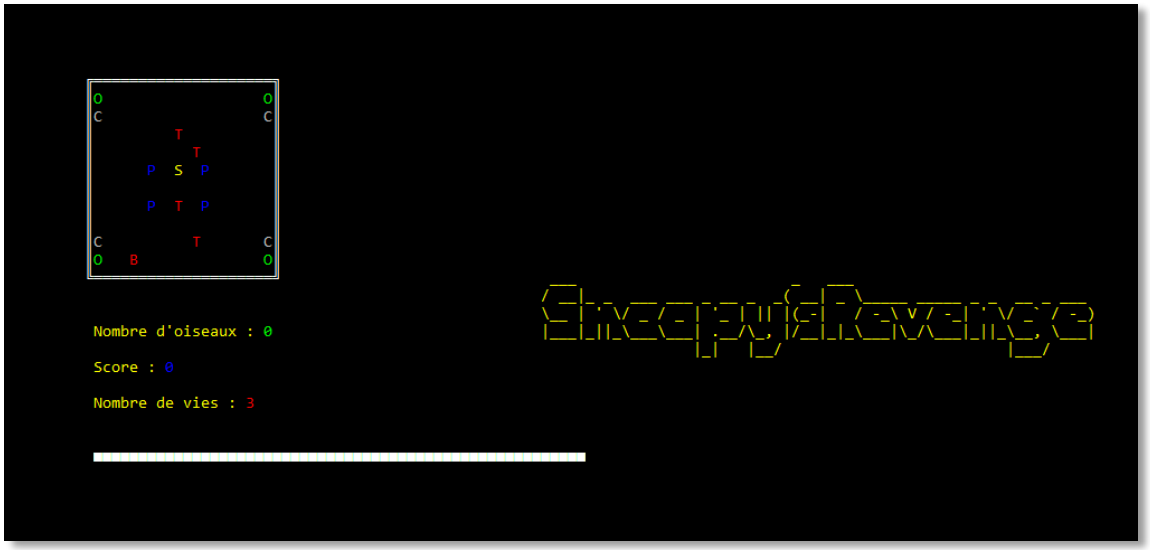
Voici le diagramme des classes de notre jeu. Il permet de montrer la visibilité des membres des classes, leur cardinalité et leur relation entre elles

Il y a une matrice pour un niveau et il y a *n* blocs pour une matrice. Le niveau et la matrice sont liés fortement (composition), les blocs et la matrice le sont aussi (relation de composition).

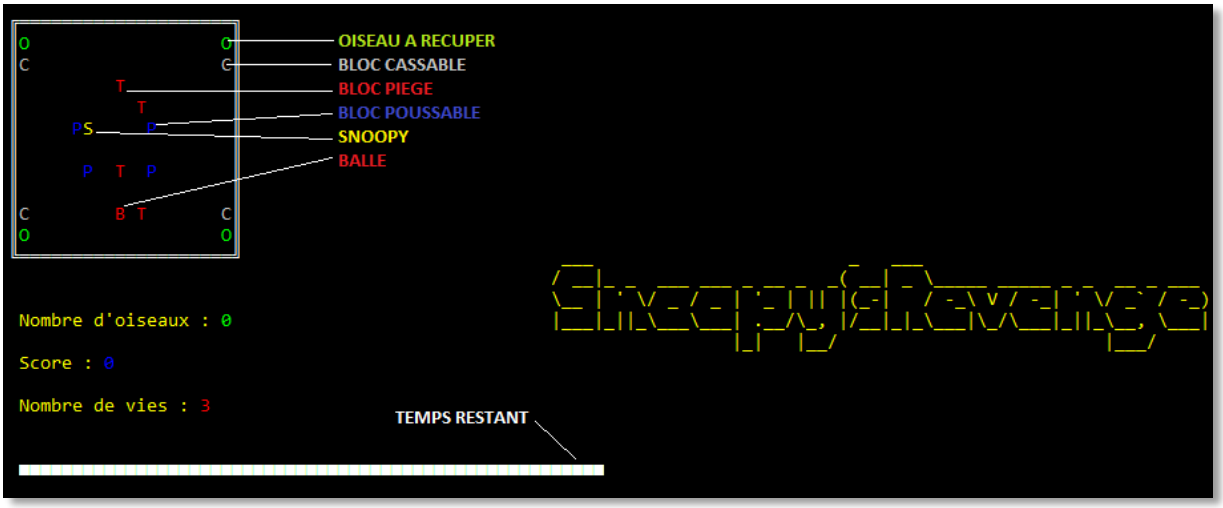
II. Scénarios

Dans cette sous-partie nous observons une partie classique de « La revanche de Snoopy ».

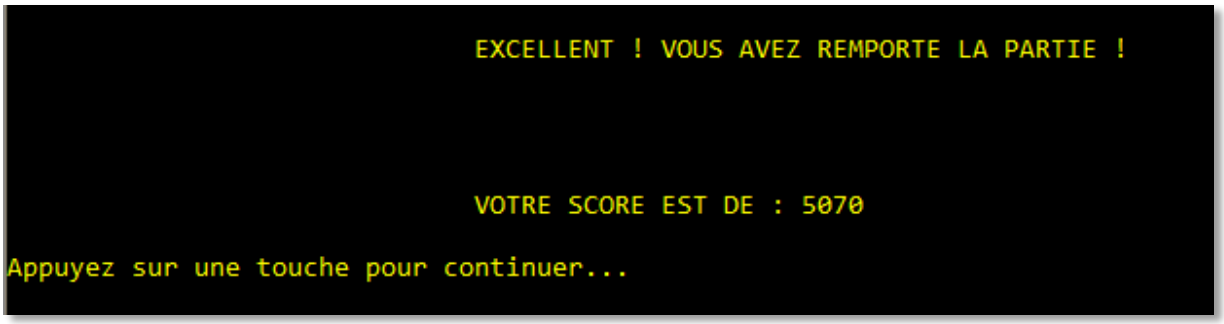
1 Le menu s'affiche à l'cran et l'utilisateur choisit une nouvelle partie



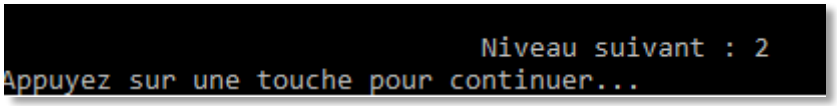
2 On entre dans le niveau 1 après une petite indications des touches utilisées



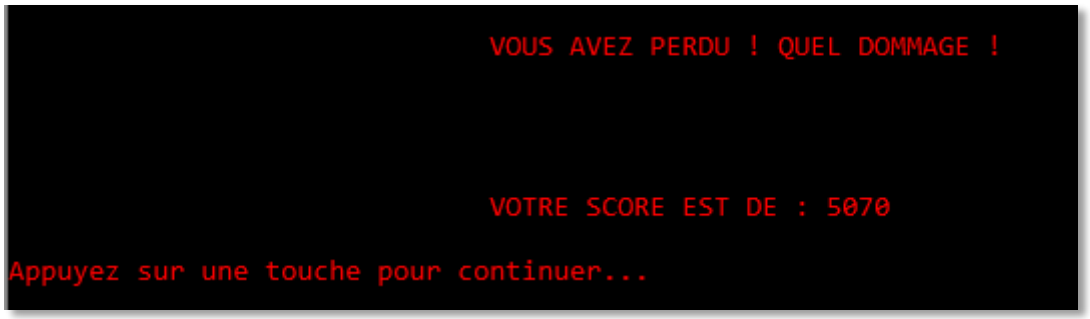
3 Schéma explicatif : La matrice contenant les différents blocs (poussables, cassables, piégés), Snoopy et la Balle.



4 Gagner la partie



5 Passer au niveau suivant



6 Ou bien perdre (quand on manque de temps ou bien qu'on a plus de vie)

On retourne au menu ensuite. On pouvait également choisir un bonus ou bien un niveau spécial en entrant un mot de passe correct.

III. Modèle et contrôleur des parties clefs du projet et découpage modulaire

1. MVC

MVC non exhaustif :

Données	Actions
Vies, score, oiseaux : entiers	Calculer, Afficher
Matrice : vecteur de vecteur de classe	Calculer, Afficher
Menu : chaînes de caractères	Afficher
Temps : double	Calculer, Afficher
Directions : caractère	Calculer
Nombre de lignes et de colonnes de la matrice : entiers	Définies par défaut
Différents blocs, le personnage : blocs	Créer, Modifier, Afficher
Choix : caractère	Saisie

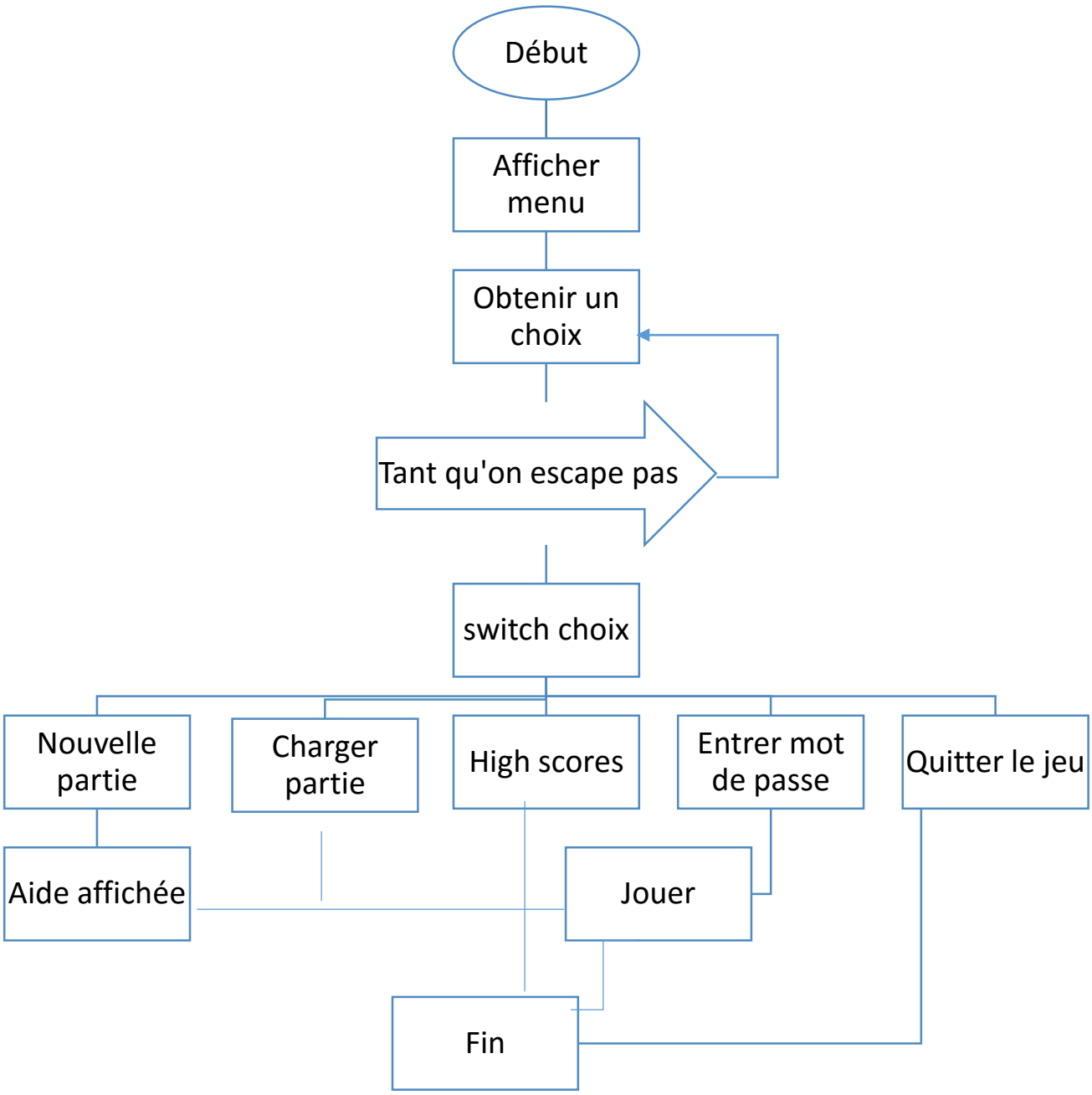
2. Analyse chronologique descendante

1. Afficher la page d'accueil du menu : avec les choix possibles (Nouvelle partie, charger partie, high scores, entrer mot de passe, quitter le jeu).

2. Obtenir le choix du joueur concernant le traitement à faire

3. Entrer dans le sous-programme sélectionné

Figure 2 Organigramme : ACD général

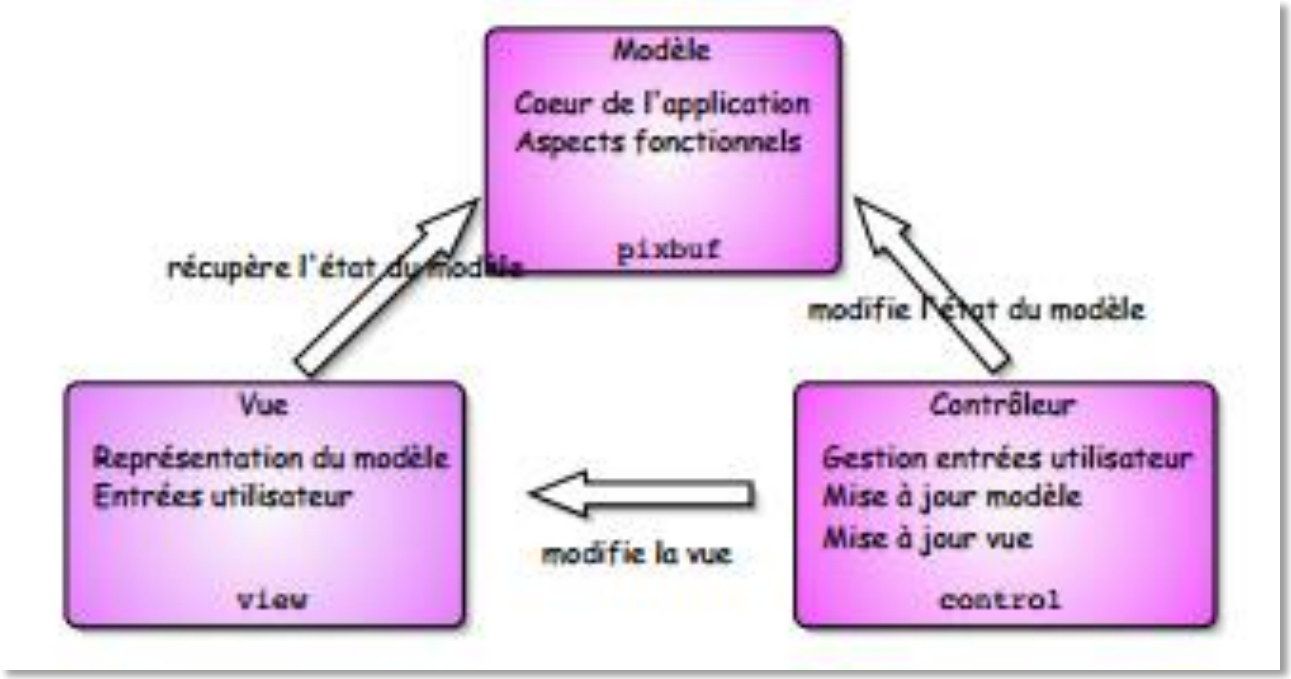


3. Découpage modulaire du projet

L'organisation de notre code permet une compréhension lisible du code :

- main.cpp : Lance le programme.
- Niveau.cpp : gère l'interaction avec le joueur (comme le menu par exemple) et met à jour le jeu selon le niveau.
- Console.cpp : gère l'aspect visuel de notre code (affichage sur la console, couleurs…)
- Matrice.cpp : gère les fonctions qui permettent la création de matrices, elle les initialise, les manipule. C'est la classe « maitresse » de notre code, le cerveau du jeu : sans elle, le reste du jeu ne tournerait pas du tout !
- D'autres fichiers.cpp correspondant à des classes qui manipulent, calculent via des méthodes.

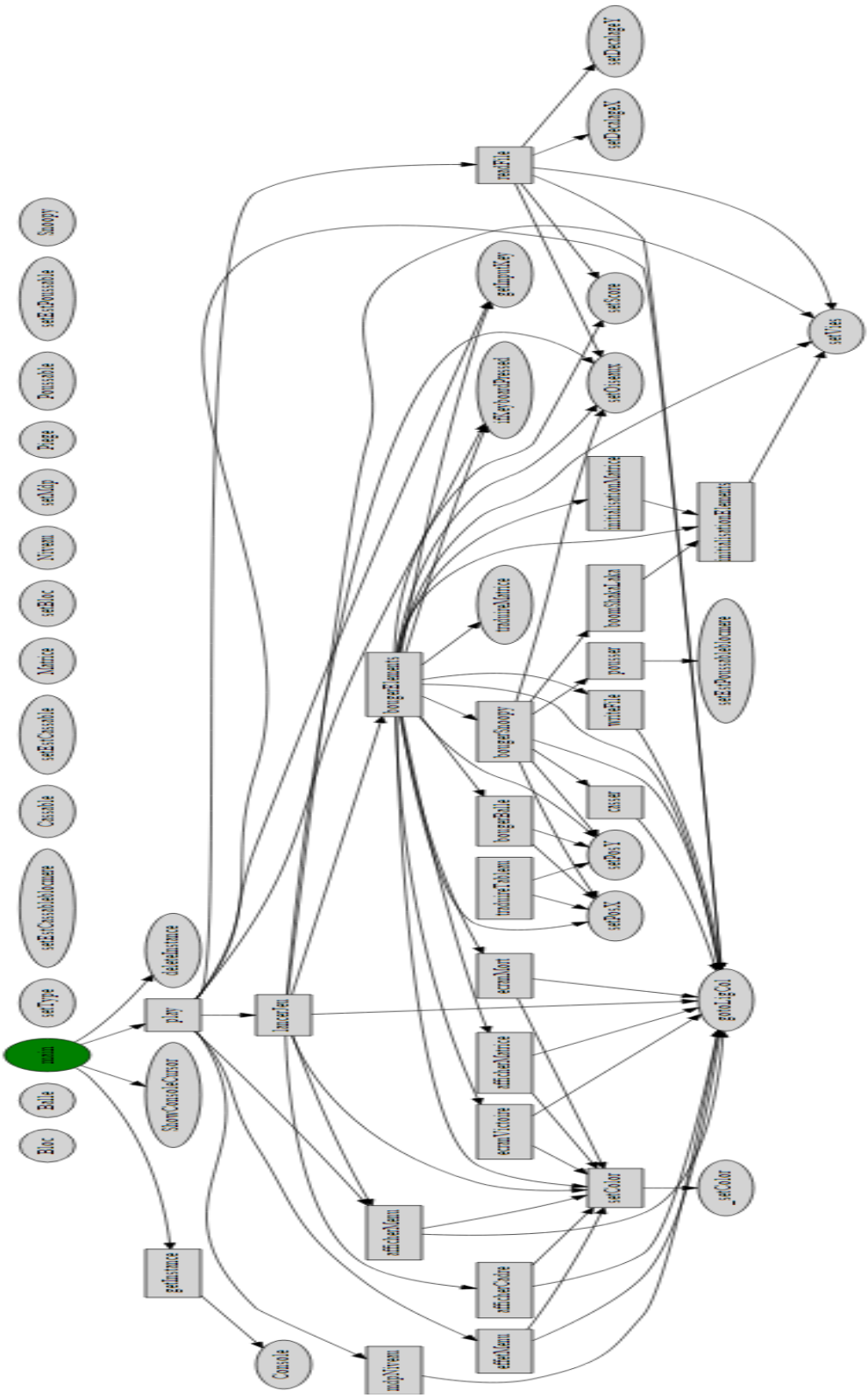
L'architecture MVC se présente sous cette forme :



http://pageperso.lif.univ-mrs.fr/~liva.ralaivola/lib/exe/fetch.php?id=teaching%3A20062007%3Aprojet&cache=cache&media=teaching:20062007:modules.pdf.

PARTIE REALISATION

I. Graphe d'appels et organigrammes de sous-programmes essentiels



Fait avec : codeanalysis.fr

II. Choix de programmation

1. Une matrice de Blocs à la place d'un tableau de chars

Le premier choix de programmation effectué en équipe fut de gérer non pas une simple matrice de chars, mais d'utiliser la puissance du langage C++ pour ne manier que des objets. Ainsi, chaque case de notre double vecteur est un bloc : vide s'il n'y rien, de type Balle si c'est la balle du jeu… La classe mère Bloc définit les attributs utilisés pour les classes filles : position X, position Y, type de bloc (retourne un « char » pour l'affichage), des booléens pour savoir si un bloc est poussable ou cassable…

2. La fonction clock(), une fonction difficile à manier

L'un des choix majeurs que nous avons fait est l'abandon de la fonction Sleep(int x) de la bibliothèque Windows.h au profit de la fonction clock(), proposée par la bibliothèque <time.h>. Cette fonction a l'avantage de permettre des tests par rapport au temps : dans notre cas, nous mesurons le temps écoulé lors du lancement d'un niveau pour faire varier le mouvement de la balle (sa vitesse).

Ce choix présente à la fois des atouts non négligeables, tout comme des points plus négatifs.

POINTS POSITIFS

Le plus important est la fluidité du mouvement de Snoopy. En effet, puisque nous ne pausons plus la boucle de jeu avec un Sleep(), fonction qui stoppe le processus de compilation pendant un certain nombre de millisecondes, chaque touche du clavier produit un effet immédiat sans le léger temps de latence autrement remarqué.

De même, à l'aide de tests simples cette fonction permet de mesurer les secondes écoulées depuis le lancement du programme (et donc du niveau) : ceci nous a permis d'afficher la barre de temps citée plus haut dans ce rapport (en affichant un espace à la place du carré précédent).

POINT NEGATIF

Celui-ci dépend surtout de notre incapacité à gérer la vitesse du processeur utilisé pour jouer : puisque celui-ci varie d'un jeu à l'autre, les boucles de tests à l'intérieur de notre boucle while() de jeu auront un impact plus ou moins important sur le rendu visuel final. Notamment au niveau du mouvement de la balle : si nous faisons un test simple (toutes les 100 millisecondes écoulées, faire bouger d'un cran), le processeur ne peux pas parcourir la boucle entière en une milliseconde, ainsi nous devons replacer le test « == 0 » par un test « <= x », x étant une valeur très petite (de l'ordre de la dizaine de millisecondes).

Ceci présente le désavantage de ne pas proposer une vitesse constante : parfois, le processeur de l'ordinateur rentrera plusieurs fois dans la boucle de test : dans ce cas-là, la balle ira très rapidement, de manière imprévisible. Parfois, au contraire, le processeur n'ira pas assez vite : le mouvement de la balle sera saccadé, moins joli.

Néanmoins, nous avons décidé d'opter pour cette option car la fluidité du mouvement de Snoopy nous semblait plus importante ! De plus, les variations de vitesse de la balle proposent une difficulté supplémentaire au joueur.

3. Sauvegarde et chargement en deux temps

Puisque nous n'arrivions pas à trouver la syntaxe sur internet nous permettant de lire un fichier contenant à la fois des chiffres et un tableau de chars, pour ensuite stocker ces valeurs dans différentes variables, nous avons décidé de séparer la sauvegarde en deux fichiers :

- Le fichier « name » qui contient la matrice
- Le ficher « name_donnees » qui contient les données du niveau : score, vies…

Lors du chargement, il y a donc deux étapes : la lecture de la matrice, puis celle des données annexes.

III. Astuces et originalités

BARRE DE TEMPS VISUELLE ET PRESENTATION GENERALE DE L'ESPACE DE JEU

La première chose qui saute aux yeux du joueur, c'est avant tout le cadre entourant la matrice et la barre de temps située en dessous : celle-ci diminue d'un carré à chaque seconde écoulée. Lorsque le temps passe en dessous de la barre des 10 secondes restantes, un message d'alerte s'affiche en dessous de la barre, informant le jouer qu'il ne lui reste plus beaucoup de temps pour terminer le niveau.

De plus, le titre de « Snoopy's Revenge » s'affiche continuellement sur la droite de la matrice, remplissant l'espace de la console pour laisser le moins de vide possible. Le menu de pause s'affiche parfaitement entre les deux, tout cela ayant été affiché à l'aide de la fonction gotoligcol().

Toujours dans l'affichage d'informations, nous proposons au joueur de voir s'afficher, sur le bord droit de la matrice, un message lui informant la touche a appuyer lorsqu'il rencontre un bloc cassable ; ce message disparait ensuite une fois le bloc cassé, ou lorsque Snoopy s'en éloigne.

Une des astuces les plus intéressantes de notre programme et le non-affichage du curseur de la console : plutôt que de voir clignoter les éléments de la matrice lorsque le curseur passe dessus rapidement à chaque affichage, nous avons trouvé sur internet une fonction capable de ne plus l'afficher, et par conséquent de réduire le clignotement général de notre jeu !

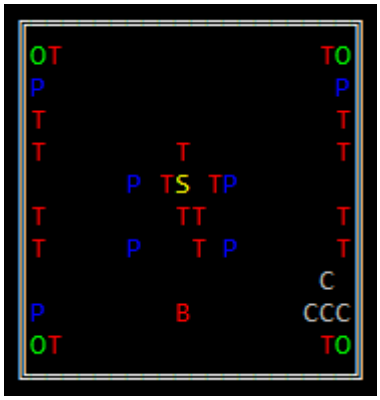
UNE BALLE QUI INTERAGIT AVEC SA MATRICE

La balle, en plus de rebondir sur les côtés de la matrice et de « manger » Snoopy, rebondit sur chaque bloc avec un système d'incrémentation de sa position X et Y en fonction d'une variable décalage : celle-ci se voit multipliée par 1 en fonction de sa collision avec un autre bloc (oiseau, bloc cassable, poussable, piégé).

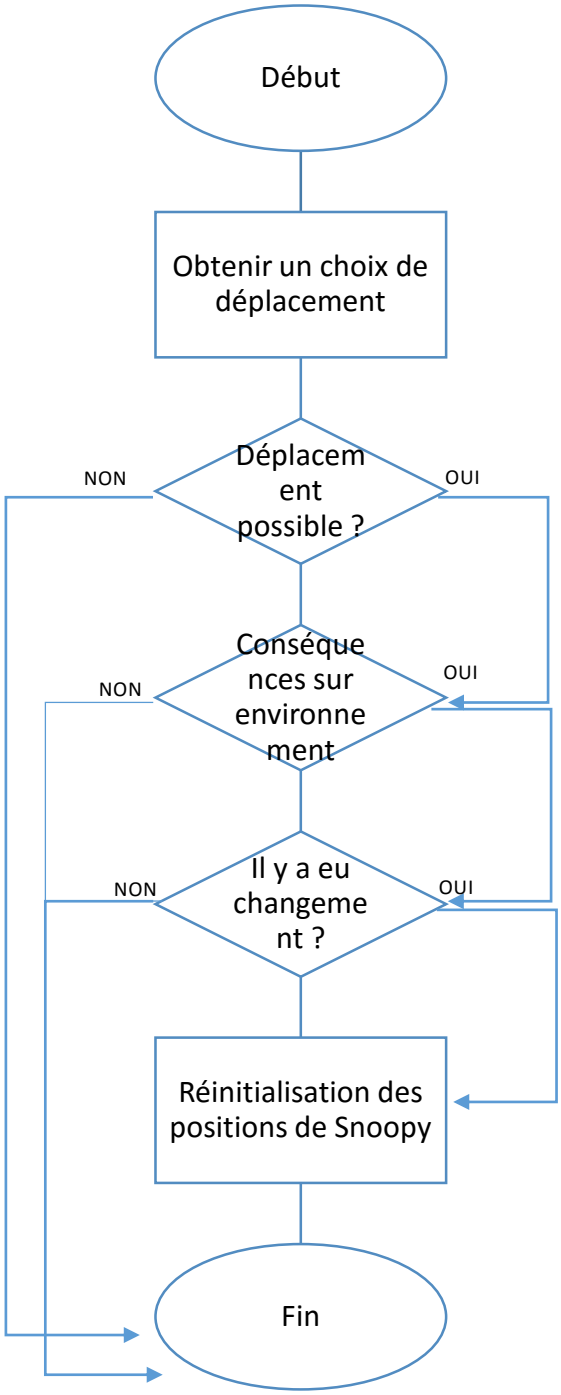
A part quelque bugs remarqués lors de sessions de jeu rapides (écrasement de blocs lorsque le processeur n'a pas le temps de mettre à jour la position de tous les blocs de la matrice et d'effectuer tous les tests), les mouvements de la balle sont assez imprévisibles pour donner au joueur une difficulté bien présente !

UN NIVEAU CACHE ET UNE TOUCHE « BOMBE » POUR GAGNER UNE VIE DE PLUS

Lors de l'entrée du mot de passe, le mot « chien » génère un niveau « caché ». Celui-ci se présente comme une matrice impossible à gagner a moins de casser les 4 blocs en bas à droite. Une fois l'oiseau obtenu, et s'il lui reste encore 3 vies, le joueur peut appuyer sur la touche « b » pour supprimer d'un coup tous les blocs de la matrice. En gagnant ce niveau, il recommence le jeu avec une vie en plus !



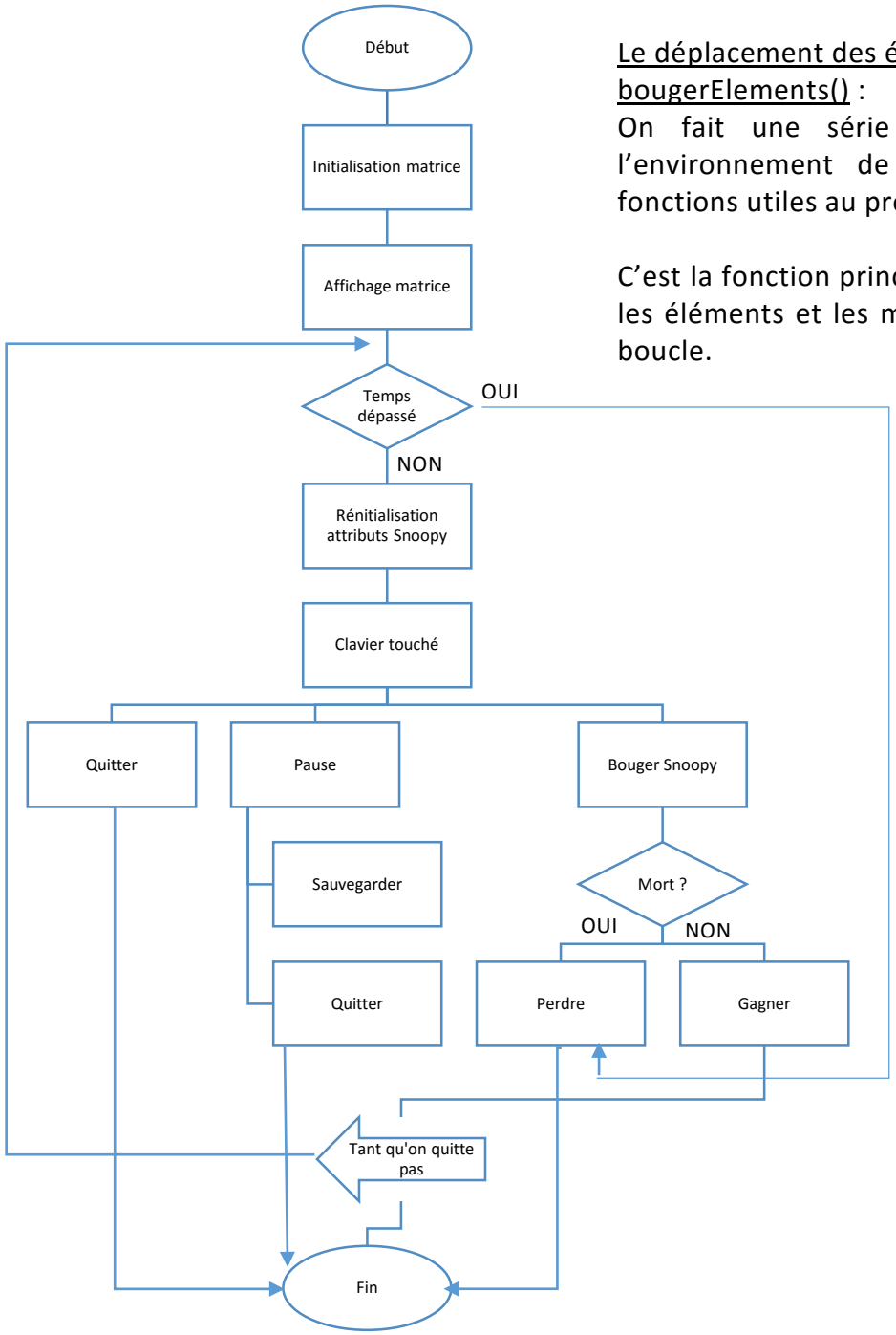
IV. Protocole de tests



Le déplacement de Snoopy : bougerSnoopy()

Cette méthode permet à Snoopy de se déplacer selon la matrice (qu'il ne peut pas dépasser) et des autres blocs qui l'entourent. Elle teste la valeur de la touche entrée par l'utilisateur pour connaître sa valeur et en déduire les actions à effectuer après avoir testé si les limites sont respectées.

Cette fonction va automatiquement appeler d'autres fonctions de la matrice, comme la fonction de cassage de bloc cassable ou de poussage de bloc poussable.



<u>Le déplacement des éléments</u> :
<u>bougerElements()</u> :
On fait une série de tests pour modifier l'environnement de la matrice et gérer les fonctions utiles au programme.

C'est la fonction principale du jeu : elle gère tous les éléments et les met à jour à chaque tour de boucle.

V. Sources

http://www.cplusplus.com/forum/
http://codeanalysis.fr/
https://openclassrooms.com/
http://stackoverflow.com/

VI. Bilan objectif collectif et individuel

Malgré une légère appréhension par rapport au choix pédagogique de placer deux ex-Prépac n'ayant jamais codé en C++ dans la même équipe, cette décision s'est révélé être bénéfique au sein du groupe. En effet, au courant de nos potentielles lacunes, nous avons su nous organiser, travailler en autonomie, chercher les connaissances et informations nécessaires pour notre projet sur Internet, dans nos cours voire même chez les professeurs.

Juliette

Ce projet m'a permis d'ouvrir mon esprit pour changer « d'orientation » : passer de la programmation procédurale à la programmation orientée objet fut un réel challenge. Il m'a aussi permis de m'ouvrir personnellement, sur le plan relationnel car chaque coéquipier est différent et travailler avec Axel fut un plaisir (très bonne cohésion de groupe et autonomie dans l'équipe).

Axel

Si clairement il nous manquait quelques semaines de préparation pour pourvoir efficacement coder ce projet, nous avons su, Juliette et moi, prendre de l'avance sur le programme pour répondre au défi posé par le sujet. Personnellement, j'ai beaucoup apprécié la programmation orientée objet offerte par le C++ par rapport à la rigidité du langage C. J'ai aussi beaucoup apprécié de travailler a deux sur ce projet (et non trois comme certains groupes), puisque ma collaboration avec Juliette a été très bénéfique !

VII. Remerciements

Nous tenons à remercier l'équipe pédagogique informatique pour les cours qu'ils ont fourni (pour l'année ING2 mais aussi de Prépac pour nos bases solides en programmation).

Nous remercions plus particulièrement M.Diedler et Mme Palasi qui ont su répondre à nos questions, nous aider à trouver nos fautes et donc à nous améliorer.