ECE
PARIS ÉCOLE D'INGÉNIEURS

Projet Informatique

GORRITY Nicolas

ROUSSEL Fabien

ING 1 – TD 05

Table des matières

I. Présentation

Le projet consiste en la programmation d'une adaptation simplifiée du jeu bien connu qu'est Farm Heroes Saga. Le jeu à développer doit être exécutable en mode console et doit comporter au moins trois niveaux de difficulté croissante.

I.1) Objectifs et règles du jeu

Une matrice de dimensions (10x15) comportant différents items vous est présentée. Le but du jeu est de réussir le dernier niveau : le niveau 4.

Pour réussir un niveau, le contrat affiché à l'écran doit être rempli en un nombre de coups limité. Il faut donc obtenir autant de points items que l'indique le contrat. Pour gagner ces points, il faut créer des combinaisons en alignant horizontalement ou verticalement au moins 3 items identiques.

Lorsqu'une combinaison est créée, elle s'efface et les items restants tombent par gravité pour remplir la matrice.
La création de combinaisons se fait par permutation des items. Un coup équivaut à une permutation. Deux items peuvent être permutés même si cela n'engendre pas la formation d'une combinaison.

Un système de comptage de points bien précis est mis en place. Pour trois items alignés, le joueur gagne trois points items. Pour quatre items alignés, le joueur en gagne huit. Pour cinq items alignes, l'ensemble des items similaires sont supprimés et autant de points sont délivrés qu'il y a d'items. Pour des combinaisons formant des L, des T, ou des croix, le joueur gagne 12, 14 ou 16 points selon les dimensions de la figure (3x3 – 4x3 ou 3x4 – 4x4).

Si le nombre maximal de coups est atteint et que le contrat n'est pas rempli intégralement, le joueur perd une vie et ne peut pas passer au niveau suivant. Ayant cinq vies au lancement du jeu, si elles sont toutes perdues, le jeu s'arrête.

Cependant, le joueur a la possibilité de récupérer toutes ses vies. Pour cela, il doit réussir un niveau. Bien sûr, si le joueur perd toutes ses vies, il les récupérera lorsqu'il relancera le jeu.

A la fin de chaque niveau, le joueur a la possibilité de sauvegarder sa progression. Il peut la récupérer plus tard à l'aide du nom qu'il aura saisi au lancement du jeu.

I.2) Ressources utilisées

Ce programme a été codé en langage C à l'aide de l'IDE CodeBlocks.

Nous nous sommes servis du service de stockage Dropbox afin de nous communiquer et de fusionner les différents sous-programmes après qu'ils aient été testés et déclarés fonctionnants.

Afin d'organiser notre planning prévisionnel nous avons utilisé le logiciel Gantt Project, téléchargeable gratuitement sur Internet.

Nous avons utilisé différentes fonctions que nous n'avions jamais étudiées en cours :

FONCTION	UTILITE	LIEN VERS LA SOURCE
ftell(fic);	Renvoie la position du curseur lors de la lecture d'un fichier texte ayant pour pointeur fic	http://openclassrooms.com/courses/apprenez-a-programmer-en-c/lire-et-ecrire-dans-des-fichiers
system(«PAUSE»);	Met l'exécution du système sur pause en attendant que l'utilisateur appuie sur une touche	http://openclassrooms.com/forum/sujet/les-instructions-comme-system-quot-pausequot-47322
Sleep(n);	Fait une pause dans l'exécution du programme longue de n millisecondes avant de le relancer automatiquement	http://openclassrooms.com/forum/sujet/les-instructions-comme-system-quot-pausequot-47322
color(flags)	Détermine selon les couleurs primaires indiquées en paramètres la couleur qui sera utilisée pour les printf ultérieurs	http://fvirtman.free.fr/recueil/04_02_01_color.c.php

Enfin, nous remercions Mr Ravaut pour nous avoir communiqué le sous-programme nommé gotoligcol.

II. Phase Conception

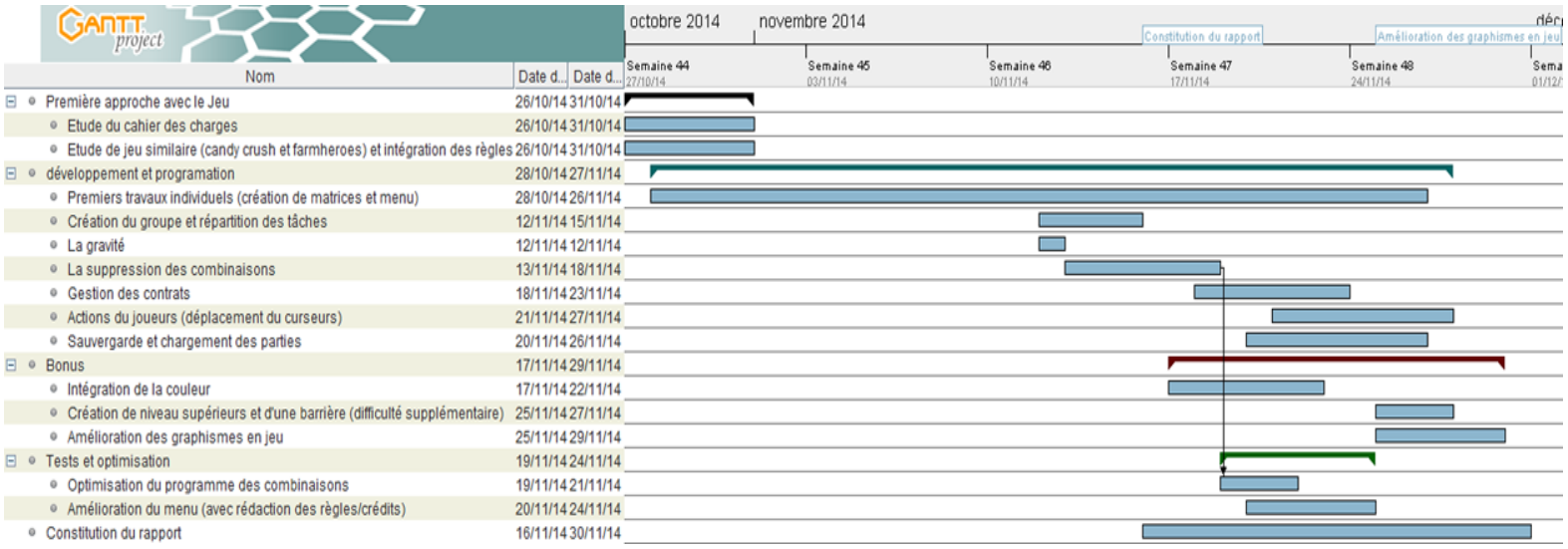
II.1) Organisation de l'équipe

Le début du travail fut d'abord individuel avant d'être collectif puisque nous avons eu le projet assez tôt et c'est seulement au retour des vacances de la Toussaint que les équipes ont pu être constituées. Ainsi, nous avions déjà commencé le projet chacun de notre côté, notamment en ce qui concerne l'analyse du cahier des charges et les premiers sous-programmes (qui constituent la base du jeu). Une fois l'équipe réunie nous avons mis en commun notre travail et cherché des solutions à nos premiers problèmes rencontrés. Nous nous sommes ensuite répartis les tâches :

- La suppression des combinaisons, le chargement de la sauvegarde, des contrats en général, les actions du joueurs pour Nicolas Gorrity.
- Le menu, la création des tableaux, la sauvegarde de la partie pour Fabien Roussel.

Une fois que l'un avait testé et terminé son sous-programme, l'autre les relisait afin de vérifier si une concision, une charge mémoire et une charge processeur plus optimale était réalisable. Dans le but de parvenir à la meilleure solution, nous avons plusieurs fois confronté notre vision des programmes et de leurs interactions.

II.2) Planning prévisionnel

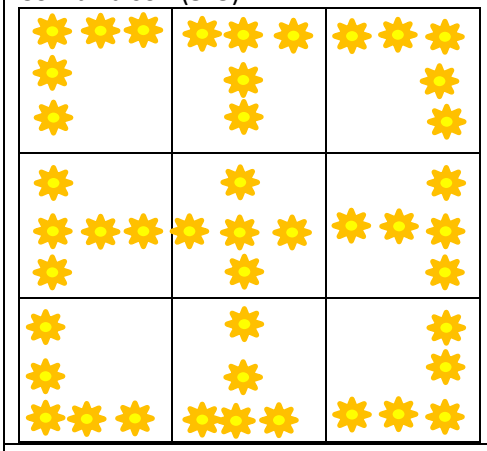
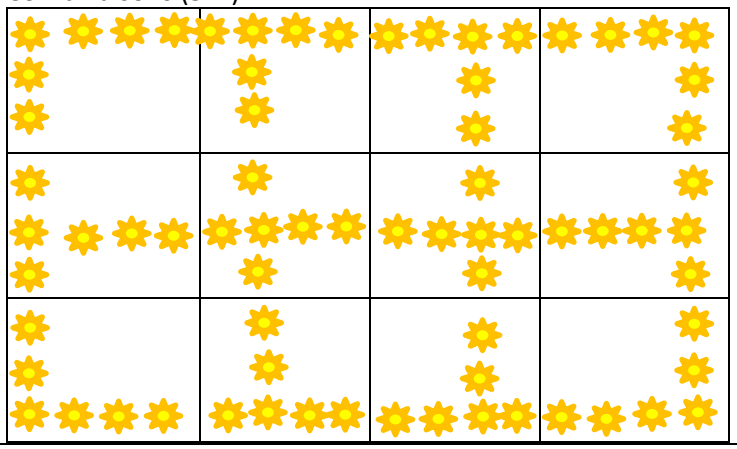


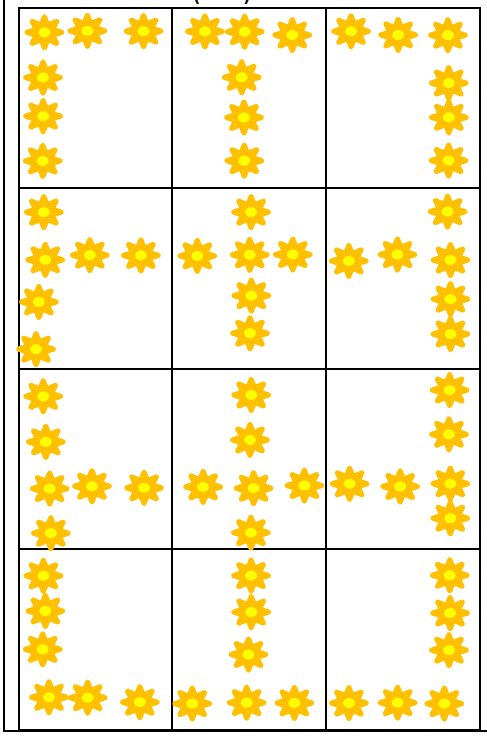
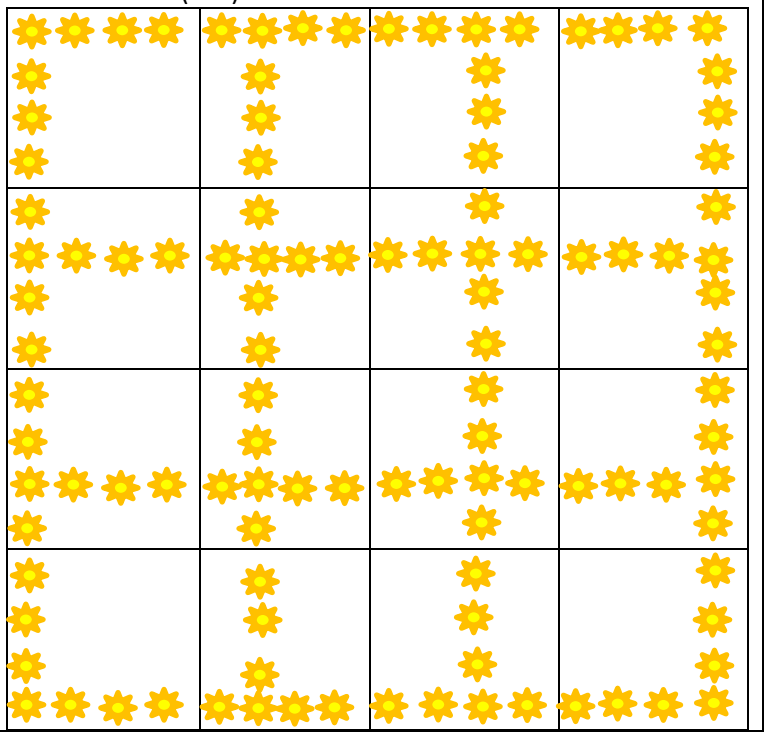
Nom	Date d...	Date d...
Première approche avec le Jeu	26/10/14	31/10/14
Etude du cahier des charges	26/10/14	31/10/14
Etude de jeu similaire (candy crush et farmheroes) et intégration des règles	26/10/14	31/10/14
développement et programation	28/10/14	27/11/14
Premiers travaux individuels (création de matrices et menu)	28/10/14	26/11/14
Création du groupe et répartition des tâches	12/11/14	15/11/14
La gravité	12/11/14	12/11/14
La suppression des combinaisons	13/11/14	18/11/14
Gestion des contrats	18/11/14	23/11/14
Actions du joueurs (déplacement du curseurs)	21/11/14	27/11/14
Sauvergarde et chargement des parties	20/11/14	26/11/14
Bonus	17/11/14	29/11/14
Intégration de la couleur	17/11/14	22/11/14
Création de niveau supérieurs et d'une barrière (difficulté supplémentaire)	25/11/14	27/11/14
Amélioration des graphismes en jeu	25/11/14	29/11/14
Tests et optimisation	19/11/14	24/11/14
Optimisation du programme des combinaisons	19/11/14	21/11/14
Amélioration du menu (avec rédaction des règles/crédits)	20/11/14	24/11/14
Constitution du rapport	16/11/14	30/11/14

II.3) Scénarii des combinaisons possibles

Afin de mieux comprendre les différentes combinaisons possibles pouvant se former à l'écran, nous avons décidé de créer un tableau ou elles seraient toutes représentées. Cela nous a aussi permis de déterminer le nombre de cas possible (55) à coder :

Ligne de 3	Ligne de 4	Ligne de 5
Colonne de 3	Colonne de 4	Colonne de 5

Combinaison (3x3)	Combinaisons (3x4)
	

Combinaisons (4x3)	Combinaisons (4x4)
	

II.4) Application de la méthode DTI

i. Les ressources utilisées

Données	Actions
tab1[10][15] : caractères	Tableau de caractère à afficher à l'écran
alea : entier	Calculer un nombre aléatoire
i, j, x, y: entier	Variables dans les boucles
r, hauteur, m : entier	Calculer
Choice, choix_fin : entier	Saisie
Vie : entier	Calculer
leveldumain : entier	Calculer
niveau_joueur : entier	Calculer
key, key2 : caractères	Saisie
Compteur : entier	Calculer
name[16] : caractères	Saisie
n_coups, nombre_coups :	Calculer

ii. Analyse Chronologique Descendante du main et du menu

0. Déclaration des variables
1. Appel de loadname pour obtenir le nom du joueur et éventuellement sa progression antérieure
2. Appel du menu
   - 2.1 Affichage des différents choix possibles
   - 2.2 Obtention du choix de l'utilisateur
   - 2.3 Analyse du choix
     - 2.3.1 Si le niveau est supérieur à 1
       - 2.3.1.1 Cas 0 L'utilisateur a décidé de jouer le niveau suivant
     - 2.3.2 Cas 1 L'utilisateur a décidé de jouer une nouvelle partie
       - 2.3.2.1 Appel du sous-programme nouveau_jeu avec niveau=1
     - 2.3.3 Cas 2 L'utilisateur souhaite consulter les règles et commandes
       - 2.3.3.1 Appel du sous-programme affichage_regles
     - 2.3.4 Cas 3 L'utilisateur a décidé de regarder les crédits
       - 2.3.4.1 Affichage des crédits
     - 2.3.5 Cas 4 L'utilisateur souhaite acheter le jeu…
       - 2.3.5.1 Affichage du troll
     - 2.3.6 L'utilisateur a fait une erreur de saisie
       - 2.3.6.1 Lancement d'un bip sonore
     - 2.3.7 Cas de la touche Echap : L'utilisateur souhaite arrêter
       - 2.3.7.1 La valeur 186 affectée au caractère choix permettra de sortir de la boucle while
   - 2.4 Tant que choix est différent de 186, que le joueur a des vies ou qu'il n'a pas terminé le niveau 4
     - 2.4.1 Réitération de 2.1 jusqu'ici

II.5) Découpage modulaire du projet

La programmation dite structurée, i.e. décomposée en différents modules, est indispensable pour un programme de cette envergure. Le code de l'exécutable final est donc divisé en plusieurs fichiers, comportant eux-mêmes différents sous-programmes.

Fichiers .o	Sous-programmes	Utilité
affichage.o	void gotoligcol	positionner le curseur à l'écran aux coordonnées transmises en paramètres
	void mettons_de_la_surbrillance	mettre l'item sélectionné par le joueur en surbrillance
	void mettons_de_la_couleur	afficher les items avec leur couleur respective
	void color	introduire de la couleur dans l'affichage à l'écran
contrat.o	void affichage_contrat	afficher l'état du contrat à l'écran en temps réel
	int def_contrat	définir le contrat de chaque niveau
Matrice.o	void gravity	faire "tomber" les items qui se trouvent au-dessus d'un espace
	void newtab	générer la matrice initiale au début du niveau
	int rang_fonction_de_item	accéder à la bonne case du tableau contrat en fonction de l'item traité dans le tableau
	int suppression_combinaisons	effacer les combinaisons du tableau, modifier en conséquence l'état du contrat, et compter le nombre de combinaisons trouvées
deplacement.o	int test_altern_fleches	permuter deux items si le joueur en a sélectionné un puis a appuyé sur 2, 4, 6 ou 8
	void action_joueur	permettre au joueur de déplacer le curseur et d'alterner deux items.
sauvegarde.o	char load_name	identifier le joueur dès le début du jeu pour une éventuelle sauvegarde ultérieure, charger son ancienne progression si elle existait
	void save_game	sauvegarder le niveau du joueur dans le fichier sauvegardejoueur.txt
menu.o	void affichage_regles	afficher les commandes et les règles du jeu
	int menu0	afficher le menu à l'écran et traiter le choix fait par le joueur
	int nouveau_jeu	gérer le déroulement d'un niveau
main.o	int main	afficher l'écran principal et lance l'affichage du menu

Tableau listant les différents fichiers objet du projet, leurs sous-programmes respectifs, et l'utilité de ceux-ci

Ce tableau présente les Entrées/Sorties de chaque sous-programme du projet :

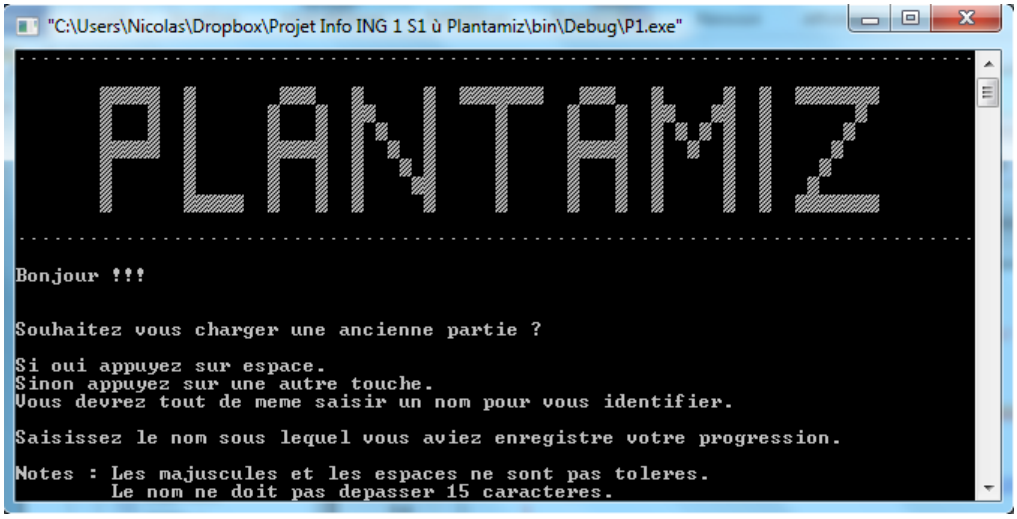
Sous-programmes	Entrées internes	Sorties internes	Entrées externes	Sorties externes
void gotoligcol	int lig, int col	/	/	affichage du curseur aux coordonnées lig,col
void mettons_de_la_surbrillance	char cha, int niveau_joueur	/	/	affichage de l'item cha avec sa surbrillance
void mettons_de_la_couleur	char ch, int niveau_joueur	/	/	affichage de l'item ch avec sa couleur
void color	int flags (couleurs primaires composant la couleur souhaitée)	couleur des textes affichés avec printf	/	/
void affichage_contrat	int contract[5]	/	/	affichage de la mise à jour du contrat
int def_contrat	int niveau, int contract[5]	modification du contrat par adresse, renvoi de int n_coups	/	/
void gravity	char tab1[10][15], int niveau	modification de la matrice de jeu par adresse	/	affichage de la "chute" des items
void newtab	char tab1[10][15], int contrat[5], int niveau	modification de la matrice de jeu par adresse	/	affichage de chaque case de la matrice
int rang_fonction_de_item	char val (valeur de la case traitée)	renvoi de int rang_tab_contrat	/	/
int suppression_combinaisons	int contrat[5], char tableau[10][15], int leveljoueur	renvoi de int compteur, modification de la matrice de jeu et du contrat par adresse	/	affichage d'espaces à la place des combinaisons trouvées
int test_altern_fleches	char val, char tab[10][15], int i, int j, int level_joueur	modification par adresse des valeurs des cases des deux items permutés, renvoi de int coup	/	/

void action_joueur	int contract[5], char table[10][15], int* pt_n_coups, int niveau	décrémentation du nombre de coups restants par adresse	commandes du joueur : espace, 2, 4, 6, 8	affichage du déplacement du curseur et du nombre de coups restants en temps réel
char load_name	int* ptlevel, char test, char name[16]	modification par adresse de name	obtention du nom saisi par le joueur, lecture des noms dans le fichier, lecture du niveau si chargement	instructions de saisie et de blindage du nom, confirmation d'identification joueur, prévient de l'erreur d'ouverture de fichier si besoin, écriture dans le fichier du nom s'il n'existait pas déjà et du niveau 1
void save_game	int level, char nameplayer[16]	/	lecture des différents noms figurant dans le fichier	écriture dans le fichier du niveau et du nom du joueur s'il n'existait pas auparavant, affichage à l'écran de la confirmation de sauvegarde ou pas
void affichage_regles	/	/	/	affichage des règles et des commandes
int menu0	char tab1[10][15], int leveldumain, char nom[16]	renvoi de int niveau au main	obtention du choix de l'utilisateur	affichage du menu à l'écran
int nouveau_jeu	char tab1[10][15], int level, int* pt_vies, char nomjoueur[16]	renvoi de int level au main, modification par adresse du nombre de vies	obtention du choix du joueur après la proposition de sauvegarde	affichage du nombre de coups initial, de la valeur du niveau, des modalités de fin de niveau (cas de victoire ou d'échec), et proposition de sauvegarde
int main				affichage de Plantamiz, proposition de chargement, affichage de fin du programme

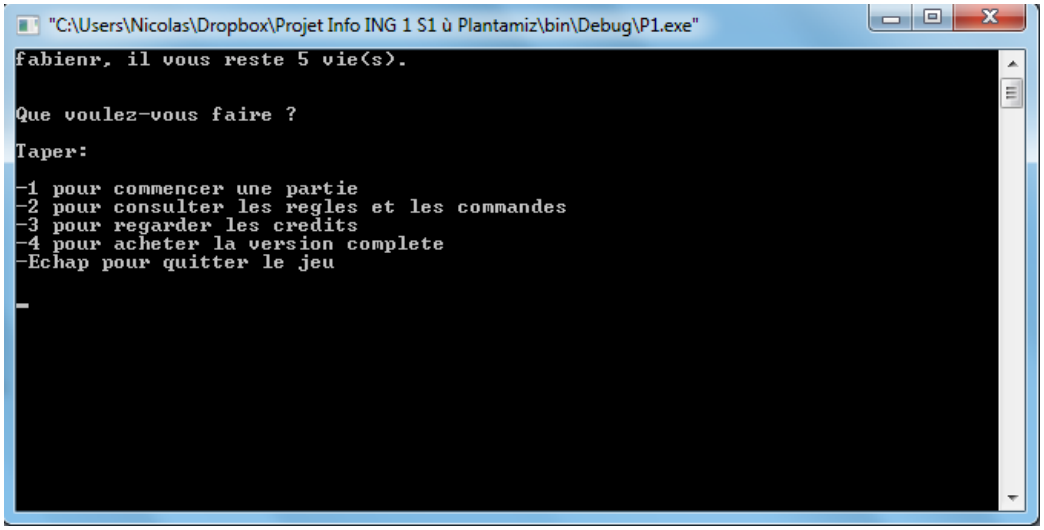
III. Phase Réalisation

III.1) Choix de programmation

Avant même d'accéder au menu, le programme demande à l'utilisateur son nom afin de lui proposer de charger une ancienne partie. Si aucune partie n'est trouvée alors, on demande à l'utilisateur s'il réessayer de charger une ancienne partie et si oui on lui redemande son nom (il est possible qu'il se soit trompé ou qu'il ait fait une faute de frappe lors de la première saisie du nom). Si le joueur ne souhaite pas charger une progression antérieure, le nom qu'il aura saisi sera enregistré dans le fichier de sauvegarde, et assimilé au niveau 1.

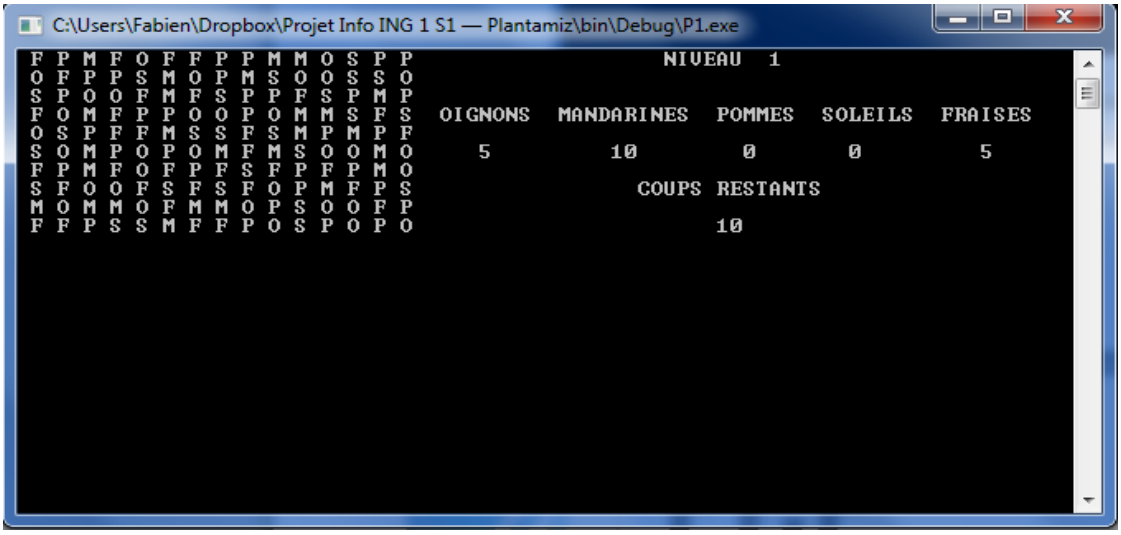


Il s'en suit le menu suivant :



Note : Malheureusement, l'option 4 n'est pas encore disponible… Pas de chance !

Lorsque nous commençons une partie, le jeu affiche le tableau aléatoirement rempli à l'écran et le vide progressivement de toutes les combinaisons éventuellement formées. Le joueur peut alors commencer sur un tableau propre. Lorsqu'il réalise une combinaison à l'écran, il peut apercevoir les phénomènes qui se passent (comme la gravité). Sur la droite il peut voir son niveau, le nombre de coups qu'il lui reste, et l'avancement de son contrat, c'est-à-dire le nombre d'items qu'il doit encore faire disparaitre pour réussir le niveau.



Le curseur ressemble à un tiret épais qui clignote sous la lettre pointée : 

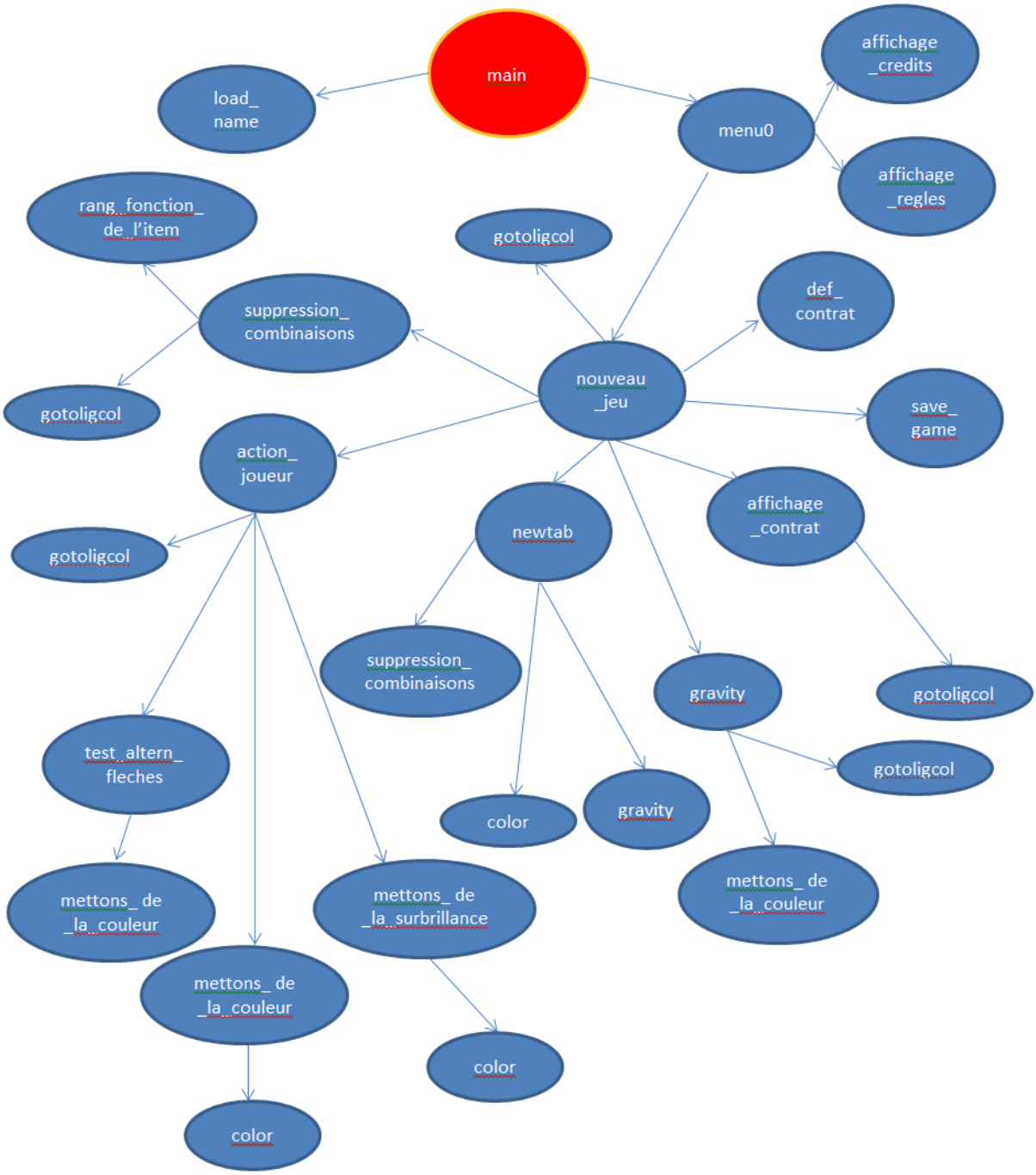
Pour le déplacer le joueur doit se servir du pavé numérique :

- La touche 2 permet de déplacer le curseur vers le bas
- La touche 4 permet de le déplacer vers la gauche
- La touche 6 permet de le déplacer vers la droite
- La touche 8 permet de le déplacer vers le haut

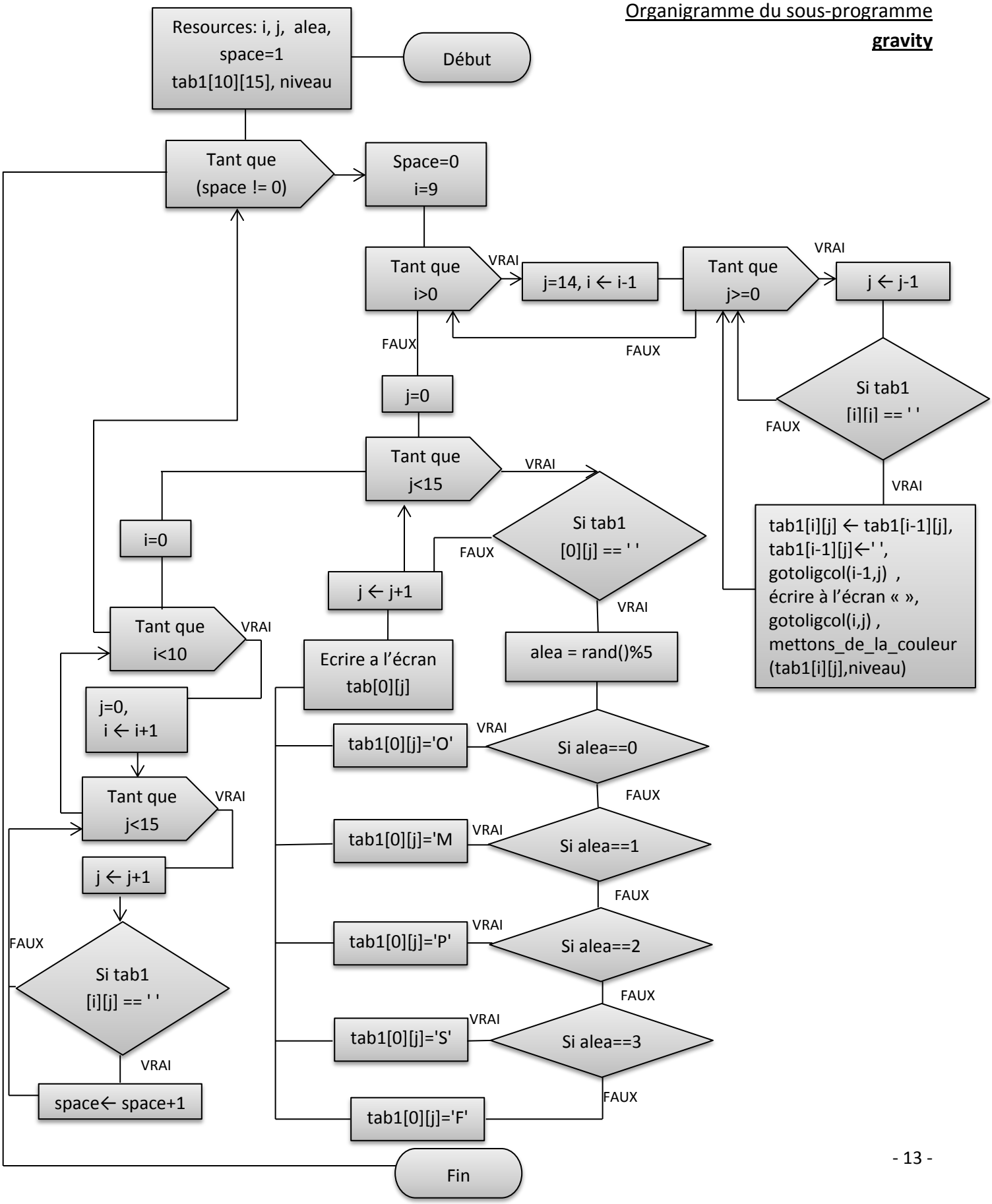
Afin de sélectionner un item, il faut appuyer sur la barre d'espacement. Puis :

- Pour le désélectionner, il suffit de réappuyer sur cette touche.
- Pour le permuter avec un autre item, il faut utiliser les touches 2, 4, 6 ou 8 avec le même principe de directions qu'auparavant.

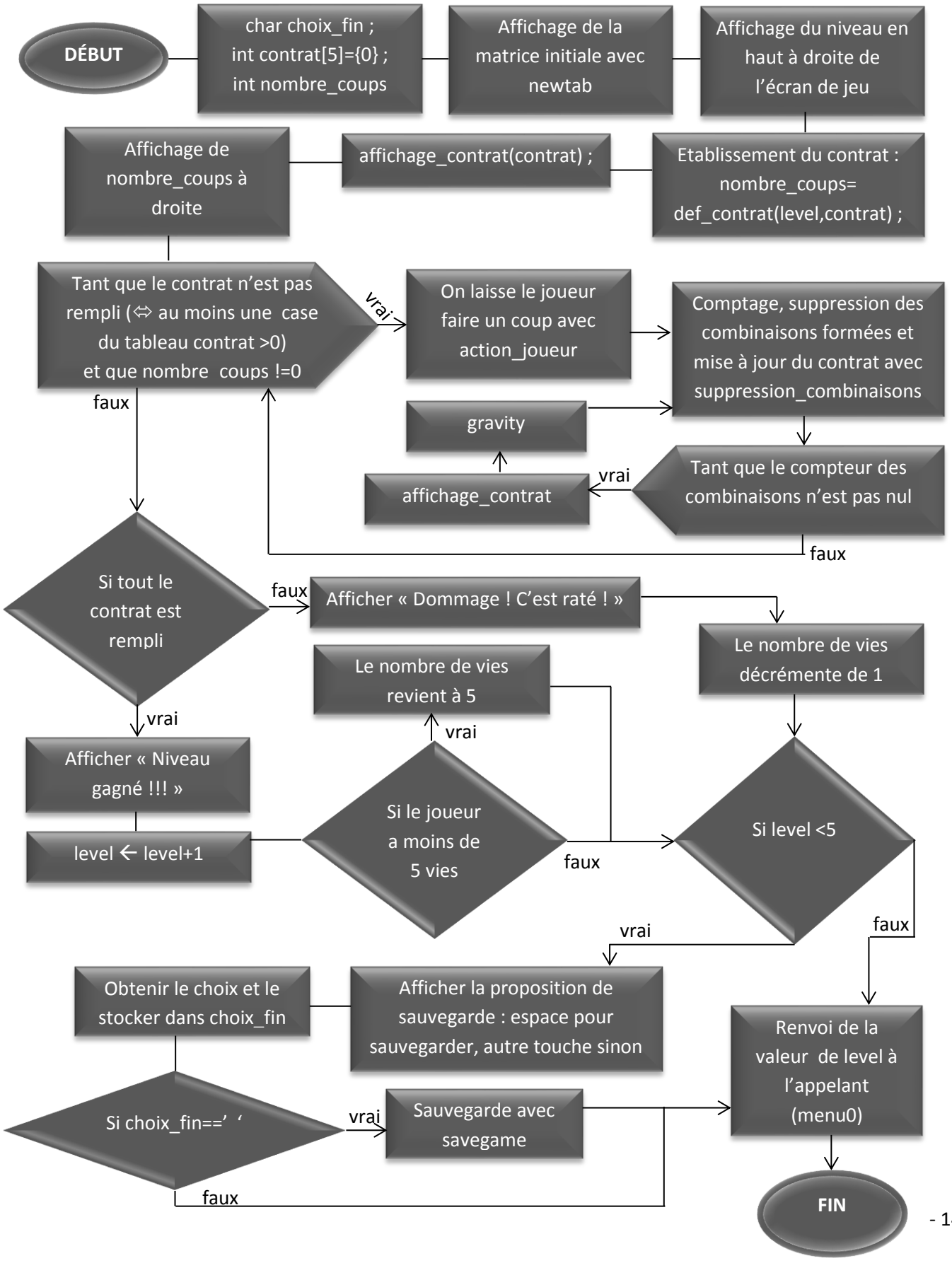
On peut ainsi résumer les interactions entres les sous programmes avec un graphe d'appel (pour plus de précision pour des interactions entre deux sous programmes, on peut se référer au tableau en II.5))



Organigramme du sous-programme **gravity**



Organigramme du sous-programme **nouveau_jeu**



III.2) Nos quelques bonus

Parmi les originalités que nous avons apportées, on compte :

- La couleur des items
  Nous nous sommes (très) rapidement rendus compte au cours des premiers tests du niveau 1 qu'il était difficile d'apercevoir les possibilités de combinaisons lorsque tous les items étaient affichés en blanc. Déchiffrer ainsi la matrice de jeu devenait donc vite désagréable et pouvait même donner mal aux yeux.
  Nous avons donc décidé d'attribuer dans les niveaux supérieurs une couleur à chaque type d'item, ce qui rendait ainsi le jeu plus agréable, mais aussi plus esthétique.
  Cependant, la fonction color ne prend en charge que les couleurs primaires (rouge vert bleu) les couleurs secondaires (cyan, magenta, jaune), et le blanc. Nous n'avons donc pas pu générer la couleur orange pour la mandarine, et avons été contraint à lui attribuer une autre couleur qui puisse se distinguer des autres, mais incohérente : le bleu.

- La surbrillance de l'item sélectionné
  Pour optimiser plus encore la lisibilité de la matrice, nous avons mis en place un système de surbrillance, grâce à la fonction color, qui s'applique lorsque le joueur sélectionne un item. La surbrillance s'enlèvera lorsque le joueur désélectionnera ou permutera l'item sélectionné.

- Le bip sonore lors de commandes erronées
  Afin de faciliter davantage la compréhension de l'utilisateur lorsqu'il joue ou qu'il parcourt le menu, nous avons instauré le lancement d'un bip sonore à l'aide de la fonction print(«\a») ; lorsque cet utilisateur appuie sur une touche qui ne figure pas parmi les commandes proposées. On peut l'entendre par exemple dans le menu, ou lors du déroulement du niveau lorsque le joueur veut déplacer le curseur en dehors de la matrice

- Le quatrième niveau
  Nous avons créé un quatrième niveau qui sera celui à réussir pour terminer le jeu. Il trouve son originalité dans le bonus qui est cité ci-après.

- Les obstacles
  Afin d'augmenter la difficulté autrement que par le biais du contrat, nous avons mis en place différents obstacles dans la matrice de jeu qui bloqueront des possibilités de combinaisons. On trouvera une barrière centrale aux niveaux 3 et 4 ; ainsi que quatre croix au niveau 4.

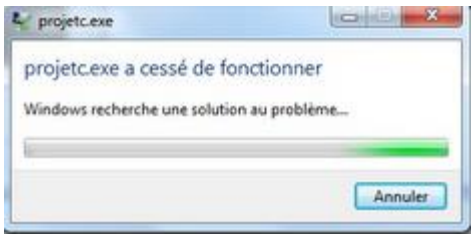
III.3) Problèmes rencontrés et résolutions

Tout d'abord, la première difficulté a été de réunir nos travaux individuels. En effet, nous n'avions pas procédé de la même façon, nous avons donc du "unifier" nos sous-programmes en prenant le plus claire et le plus synthétique.

Ensuite, nous avons passé du temps avec la suppression des combinaisons : comment faire un sous-programme qui puisse être à la fois le plus concis mais aussi le moins répétitif ? Cette question nous a permis d'arrivé à un bon résultat plus optimisé que nous l'avions imaginé grâce à un travail de préparation rigoureux sur papier.

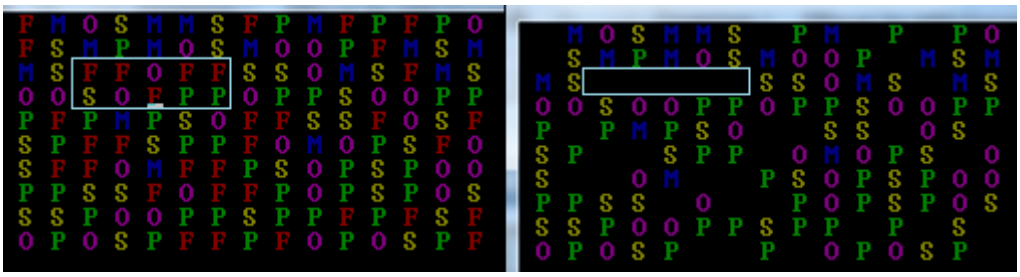
Par la suite nous avons été confrontés à un obstacle que nous n'attendions pas à rencontrer : la sauvegarde. En effet il fallait pouvoir écrire et lire en même temps dans le fichier, on avait donc opter pour une ouverture en "a+" puis un « fseek(fichier, 0, SEEK_SET); » pour revenir au début du fichier. La solution semblait marchée sur le papier mais le jeu « cessait de fonctionner » ou ne sauvegardait pas la partie. Nous avons donc cherché sur internet, et puis le site openclassrooms.com nous a proposé ftell(fic). De cette manière, nous avons élaboré la solution suivante : nous commençons par ouvrir le fichier avec les droits de lecture, nous récupérons la position du curseur (avec ftell) après avoir trouvé le nom, nous fermons le fichier et le rouvrons avec les droits "r+" puis on se place sur la position enregistré, on y ajoute le niveau et enfin on ferme le fichier.

Enfin le dernier problème que nous avons rencontré a été le son. Nous voulions introduire de la musique et des bruitages en jeu (qu'on aurait pu désactiver) et nous avions donc téléchargé la bibliothèque FMOD (http://openclassrooms.com/courses/apprenez-a-programmer-en-c/jouer-du-son-avec-fmod). Mais on a eu quelques problèmes : l'un de nous ne pouvait pas accéder au fichier et lorsqu'on avait enfin pu contourner le problème, lors de la compilation plusieurs messages du type « C:\Documents\P1\main.c|13|undefined reference to `FMOD_System_Create@4'| ». Et nous n'avons jamais pu lancer le jeu sans avoir d'erreur de compilation ou sinon le jeu qui se crashait au lancement. Par manque de temps nous n'avons pas pu trouver de solution à cet obstacle.



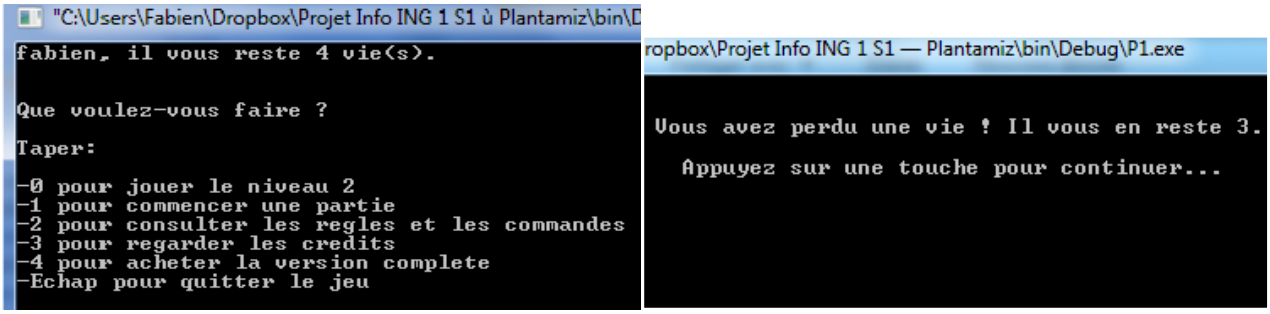
III.4) Protocoles de tests

- Ligne ou colonne de cinq items identiques



Lorsque 5 items identiques sont consécutivement alignés, on a l'ensemble des items identiques à ceux alignés qui s'effacent et qui laisseront place à la gravité. Dans ce cas de figure, la permutation aura rapporté 35 points Fraise au joueur étant donné qu'il y avait 35 items Fraise présents dans la matrice.

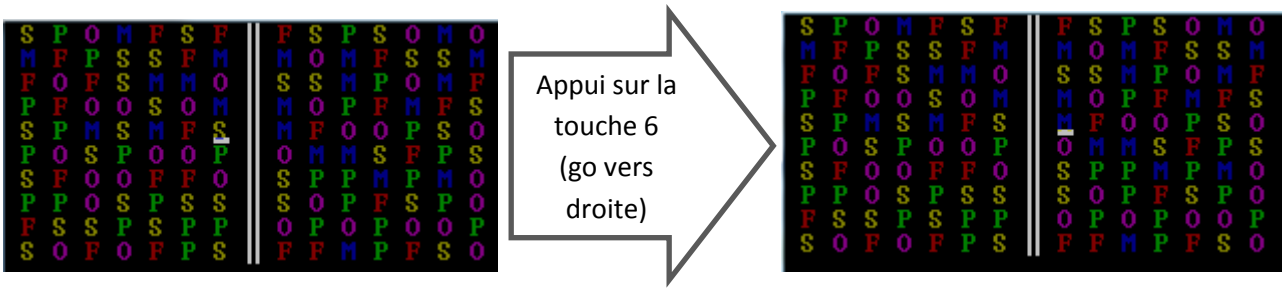
- Décompte du nombre de vies

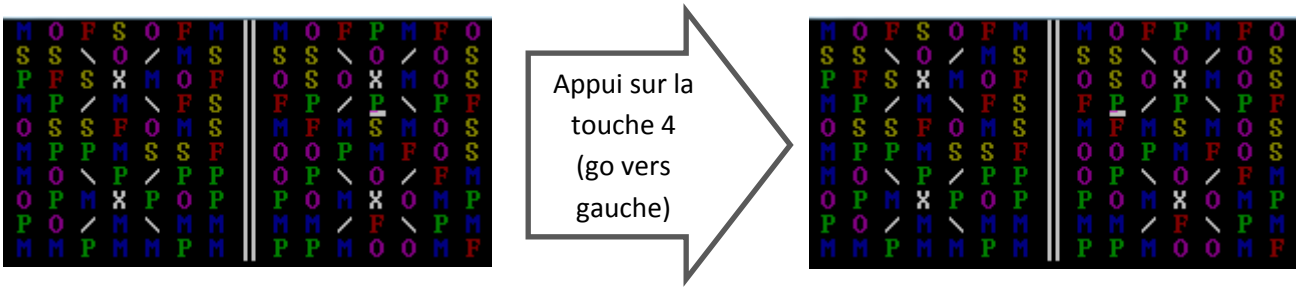


A chaque affichage du menu, et à chaque fois que le joueur perd un niveau, le nombre de vies qu'il lui reste est clairement affiché à l'écran. Si le nombre de vies tombe à zéro, le jeu s'arrête.

- Blindage du déplacement du curseur sur les obstacles

Les obstacles aux niveaux 3 et 4 ont nécessité un blindage pour empêcher le joueur de pouvoir les permuter avec d'autres items.





Si le joueur tente de déplacer le curseur vers un obstacle, cet obstacle sera détecté et le curseur « sautera » cet obstacle pour arriver une case plus loin.

Cependant, si le joueur sélectionne un item et tente de le permuter avec une partie d'un obstacle, rien ne se passera. L'item sera simplement désélectionné.

IV. Bilan

IV.1) Conclusions collectives et personnelles

Bilan collectif :

Nous sommes très satisfaits de rendre un projet respectant intégralement le Cahier Des Charges proposé par l'équipe enseignante, et ce dans les délais imposés. Nous aurons appris à gérer le temps qu'il nous était imparti ainsi que les contraintes du Cahier Des Charges.

Nous sommes également fiers d'avoir même pu proposer des petites originalités supplémentaires qui étaient autorisées dans tout autre niveau que le premier. Nous avons tous les deux appris une nouvelle fonction : ftell.

Bilans personnels :

- Nicolas Gorrity : Ce projet informatique n'est pas le premier que j'aie réalisé. L'an dernier, ayant fait la spécialité ISN en Terminale, j'ai eu l'occasion de participer à un projet à huit personnes en tant que chef de projet, également sur un jeu, mais sur une durée de plusieurs mois et sous le langage Python.
  Le projet que nous venons de réaliser a été une expérience totalement différente de celle que j'ai pu vivre l'an dernier, aussi bien sur le fond que sur la forme. Le plus gros contraste est sans aucun doute la rigueur de l'organisation, du travail papier et de la conception avant de commencer le code.
- Fabien Roussel : Etant débutant dans le domaine de l'informatique, je pense que nous avons conçu une solution viable mais optimisée. Nous avons rapidement compris les enjeux du projet ce qui nous a permis d'avancé et bien. Ce premier projet m'as permis de mobiliser et relier toutes les connaissances que j'ai acquises cette année, d'avoir une première approche des travaux de groupe et ainsi de me motiver pour la suite.

IV.2) Les améliorations possibles

- Mise en place de la musique et/ou de petits fonds sonores (ex : sons de victoire, de défaite)
- Possibilité de faire jouer un second joueur
  - Mode Duel : le premier qui remplit le contrat gagne
  - Mode Coop : chaque joueur peut permuter un item par coup → deux paires d'items peuvent être déplacées à chaque coup → contrats très difficiles