Rapport Projet Jeu de Paires

Introduction :

Notre projet est basé sur la programmation à l'aide d'une interface graphique du Jeu de Paires. Celui-ci devait être simple et contenant un choix de difficulté changeant par conséquent la taille de la grille du jeu. Nous avons choisi d'inclure dans celui-ci à nos dépends des thèmes permettant à l'utilisateur de se rapprocher d'un environnement plus agréable et correspondant à ses goûts.

Fonctionnalités du programme :

Les fonctionnalités de notre programme sont multiples et variées. L'utilisateur aura la possibilité de sélectionner via l'un des menus le thème de son choix puis la difficulté. Le programme, affichera ensuite son choix de thèmes, pour identifier celui-ci il appliquera la valeur de retour des fonctions de thèmes et des difficultés, sur le chemin d'accès modifiable dans la fonction snprintf (bien plus claire via les images suivantes).

```
struct renvoie s_difficile = {3, "D"};
return s_difficile;
```

Ex dans le cas de struct dbd (même chose pour sao à la différénce struct_sao ;

```
struct renvoie struct_dbd = {1, "D"};
return struct_dbd;
```

```
/* On retourne l'image sur la quelle le joueur a cliqué */
snprintf(save,50,"jeu/im_%s/0%d.png", theme->valeur, tableau[loc_tab_x][loc_tab_y]);
ChargerImage(save, pos_X-150, pos_Y-150, 0, 0,100, 139);
```

L'utilisateur aura accès à un timer qui lui donnera une visibilité sur le temps écoulé depuis le début de sa partie en haut à droite de l'écran. Qui s'arrêtera en fin de partie lui donnant son temps de jeu avant de gagner.

```
/* TIMER */
var= Microsecondes();
unsigned long cycle = 1000000;
char str[70];
couleur coo;
couleur coul;

/* ================================================
/* BOUCLE INFINI */
/* ================================================

coo=CouleurParNom("blue");
coul=CouleurParNom("black");
while(0!=1){
    ChoisirCouleurDessin(coo);
    DessinerRectangle(950,130,200,30);
    RemplirRectangle(950,130,200,30);
    var2=(Microsecondes()-var)/cycle;
    snprintf(str,20,"Temps de jeu : %ld", var2);
    ChoisirCouleurDessin(coul);
    EcrireTexte(960,150,str,1);
```

Un mode tricheur est aussi existant. Le joueur n'aura qu'a appuyer sur sa touche « t » (majuscule ou minuscule peu importe). Celui-ci donne accès à l'ensemble des cartes, lui permettant donc de mémoriser tous les pairs du jeu. Il est toutefois important de noter que dans notre memory le mode tricheur efface toutes les pairs trouvées (toutes les cartes sont faces retournées à la reprise).

```
if(ToucheEnAttente()==1){
   if(((Touche()==84) || (Touche()==116)) && (check_t==MODE_CHT_DESACT)){
      check_t=MODE_CHT_ACT;
      longueur=90;
      hauteur=80;

      taille_x=6;
      taille_y=4;

      if (difficulte->type_renvoi != 3){
         longueur += 150;
         taille_x = 4;
         if(difficulte->type_renvoi == 1){
            taille_y = 3;
         }
         else{
            taille_y = 4;
         }
      }

      for(j=0;j<taille_y;j++){
         for(i=0;i<taille_x;i++){
            longueur +=150;
            snprintf(crocs,50,"jeu/im_%s/0%d.png", theme->valeur, tableau[i][j]);
            ChargerImage(crocs, longueur, hauteur, 0, 0,100, 139);
```

L'élément le plus important reste le jeu en lui-même. En effet notre programme suit les règles classiques de la memory : Il doit cliquer sur deux images, si celles-ci ne se correspondent pas alors après un cours laps de temps celles-ci repassent en face cachées. Toutefois si les deux images sont des pairs alors elles resteront de face. Une fois l'ensemble des pairs visibles, le jeu prend fin affichant un menu de fin.

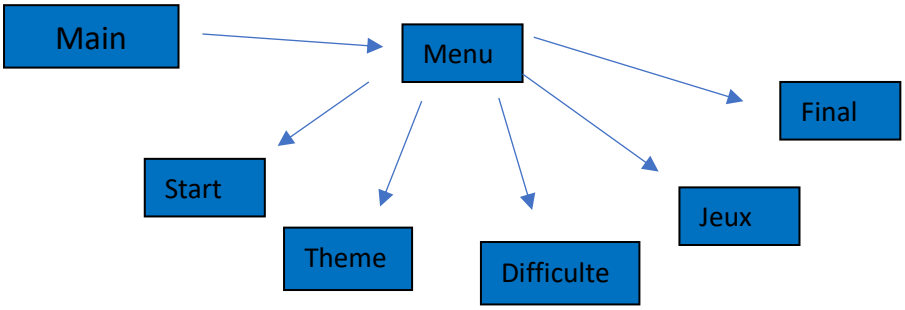
Structure du Programme :

L'ensemble de notre jeu est divisé sur plusieurs fichiers qui s'appellent entre eux. Et qui adoptent une logique cohérente à l'affichage graphique des différents onglets. On retrouve tout d'abord « main.c » qui est l'étape de lancement du jeu elle appelle la fonction « menu » qui est le hub de chacun des autres fichiers. Menu appelle tout d'abord le fichier « start » contenant l'interface graphique du menu, dans celui-ci se trouve un bouton start qui une fois pressé renvoie vers Menu qui appellera le fichier suivant « theme ». Dans thème l'interface graphique affichera deux choix à l'utilisateur et en fonction de son clic il renverra une valeur dans la structure du thème choisi (voir capture pour plus de compréhension).

```
/* Bouton choix SAO */
if(_X>=200 && _X<=480){
        if (_Y>=250 && _Y<=1850){
                struct renvoie struct_sao = {1, "S"};
                return struct_sao;
        }
}
/* Bouton choix DBD*/
else if(_X>=800 && _X<=1094){
        if (_Y>=250 && _Y<=1850){
                struct renvoie struct_dbd = {1, "D"};
                return struct_dbd;
        }
 } if(_X>=50 && _X<=350){
         if(_Y>=200 && _Y<=300){
                 struct renvoie s_facile = {1, "F"};
                 return s_facile;
         }
    } if(_X>=50 && _X<=350){
           if(_Y>=350 && _Y<=450){
struct renvoie s_moyenne = {2, "M"};
                 return s_moyenne;
         }
    } if(_X>=50 && _X<=350){
           if(_Y>=500 && _Y<=600){
struct renvoie s_difficile = {3, "D"};
                 return s_difficile;
         }
```

Theme renvoie à nouveau au fichier « menu » qui appellera cette fois « difficulte », offrant à l'utilisateur 3 choix de difficultés, une fois celle-ci choisie par l'utilisateur,le programme renverra une structure correspondant à la difficulté.

Enfin, après avoir renvoyer a l'utilisateur la structure correspondante à la difficulté, le programme appellera de manière consécutive les fichier jeux et final. Toutefois le fichier final ne s'éxécutera qu'après la fin de jeux (c'est-à-dire lorsque l'utilisateur aura trouvé toutes les paires). Nous pouvons donc voir que chaque fichier correspond à un onglet de notre jeu, Final fait référence à l'écran de victoire.



Explication des données :

Les variables utiles à la représentation de la grille sont nombreuses on y trouve :

Lolea et Lalea → variables longueur et largeur aléatoire stockant les emplacements tableaux ou seront affichés chacune des images.

Acc → (en lien avec les variable précédentes) variable stockant les numéros d'image et se décrémentant touts les deux tours d'images afin de stocker les numéros dans le tableau.

taille_x, taille_y → stockant la taille abscisse et ordonnée du tableau. Elle varie en fonction de la difficulté.

pos_X, pos_Y → variables servant dans les tests de cliques utilisateur. S'incrémentant de la taille de l'image lorsque celle-ci ne correspond pas au clic du joueur.

loc_tab_x et loc_tab_y → variables stockant l'indice correspondant a la zone ou l'utilisateur à cliqué (en lien avec pos_X et pos_Y). En effet celles-ci s'incrémentent en même temps que pos_X et pos_Y lorsque le clic n'est pas au bon endroit.

distMnX, distMxX, distMnY, distMxY → variables qui indique lors de l'exécution de l'algorithmes si la zone de clic ne dépasse pas la taille de la grille. Elle stock les extrémités de la grille en termes de coordonnées.

tableau[ ][ ] → est un tableau stockant les valeurs en Int des images. C'est la représentation brute de la grille graphique. Elle permet à l'origine de stocker chaque image et d'être utilisé plus tard en termes de comparaison.

Exposition de l'algorithme :

L'algorithme d'affichage du tableau est en 2étapes :

-Tout d'abord il y a l'affichage des cartes retournées, en partant des coordonnées pos_X et pos_Y qui sont initialisées en fonction de la difficulté choisie. Le programme affichera en deux boucles for les images une par une. Il affichera l'image puis avancera de la coordonnée de l'image +50 et au bout de 4 ou 3 affichages d'images (en fonction de la difficulté) passera à la ligne grâce à la première boucle.
Dans la variable tableau il y a avant tout un remplissage de valeur -1. Celles-ci seront utilisées pour identifier les emplacements vide. Après cela le tableau est rempli sur des emplacements aléatoires avec les numéros d'images, en effet des int remplissent les

cases du tableau entre 0 et 12, ce qui permettra plus tard grâce à la fonction snprintf de faire le lien avec nos images qui sont nommés de 00.png à 011.png.

-L'étape suivante correspond au test de clic. On lance une boucle infinie dans laquelle un test de clic est répété. Une fois celui-ci accompli on récolte les coordonnées utilisateur dans les variables _X et _Y si les clics sont au-dessus de distMxX, distMxY ou en dessous de distMnX, distMxX alors le test est rejeté. Si ce n'est pas le cas un test est effectué en partant de la coordonnée extrême x gauche et y gauche. Si le test entre la coordonnée actuelle de (pos_x et pos_x + coordonnée image) est fausse alors la coordonnée loc_tab_x s'incrémente de 1 pour indiquer qu'on avancera d'un indice aussi dans le tableau multidimensionnel, on incrémente aussi pos_x de la coordonnée de l'image puis on effectue le test de coordonnée entre pos_x et pos_x+50 si le test s'avoue positif, on annule tout car cela signifie que l'utilisateur a cliqué entre deux images, dans le cas contraire on réeffectue nos test précédents après avoir incrémenté pos_x de 50.

Une fois la bordure atteinte et la coordonnée toujours pas encadrée, on réeffectue le tout en incrémentant pos_y de 30 et loc_tab_y de 1. On ajoute cette fois ci un test entre pos_y et pos_y -30 afin de tester si l'utilisateur n'a pas cliqué au-dessus de l'image.

Pour finir dès que notre algorithme est exécuté et la coordonnée retrouvée on peut identifier l'image sur laquelle l'utilisateur à cliquer grâce à loc_tab_x et loc_tab_y qui nous serviront d'indice pour aller chercher notre image dans le tableau (tableau[loc_tab_x] [loc_tab_y]). On affiche ensuite grâce à snprintf l'image correspondante à la valeur dans le tableau.

Conclusion Personnelle

Samy Meliani :

Selon moi ce projet à eu des aspects positifs comme négatifs. En effet grâce à celui-ci j'ai pu me mettre directement dans le bain et voir à quel niveau de difficultés et à quel niveau d'organisation j'allais devoir m'habituer, et s'il y a quelque chose que j'ai pu en tirer, c'est que ça n'a rien à voir avec les projets de NSI de par l'écart d'investissement personnel que cela demandait. Pour ma part je relève quelques points négatifs, comme le manque d'organisation pour le projet et surtout l'investissement dans celui-ci étant donné que nous aurions du nous y prendre dès l'annonce du projet et pas 2 semaines après. Néanmoins mon camarade et moi restons tout de même assez fier de notre production et y avons appris beaucoup.

Fauvet Matthis :

Ce projet était vraiment exceptionnel et pour plein de raison. Tout d'abord, j'ai pu mettre en application une grande partie de ce que nous avons travailler en cours. De plus, devoir travailler avec un équipier m'a enseigné le travail d'équipe que je ne connaissais pas réellement puisque je travaillais tout le temps en solo l'année précédente. Également, ce travail ma permis de travailler tout ce que je ne comprenais pas ou bien ce que je n'appliquais pas l'année dernière comme l'optimisation du code, ou m'intéresser à comment la machine va interpréter la demande que je vais lui faire.

Enfin, cette SAE a été très importante pour moi puisqu'elle m'a fait comprendre que je vais devoir mettre les bouchées doubles pour continuer dans la voie de l'informatique. En effet, puisque je faisais partie des meilleurs de mon établissement lycée dans la matière NSI, je pensais que j'allais réussir cette année facilement. Cependant, je me rends compte aujourd'hui que je ne suis pas plus fort qu'un autre et que si je ne veux pas me faire larguer dans la foule, je vais devoir redoubler d'efforts.

En conclusion, que ce soit du côté programmation en C, programmation en équipe ou réflexion, cette SAE n'a eu que des effets positifs sur moi.