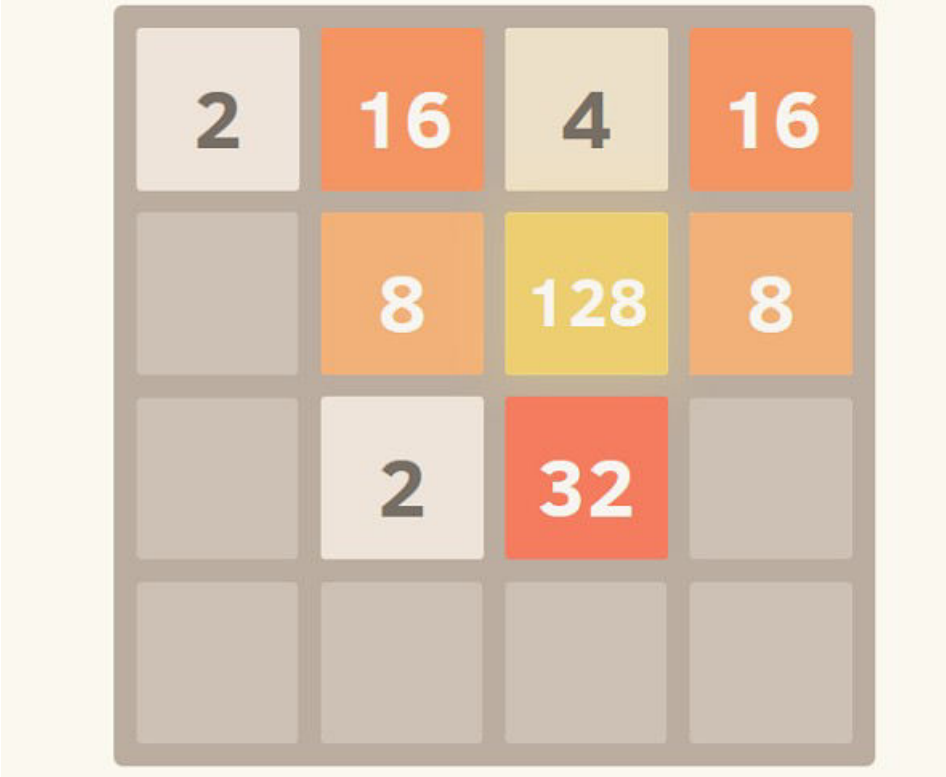
PROJET D'INFORMATIQUE

Le jeu 2048



LABADIE *Yannis* & BEKKARA *Mohamed*

CY TECH

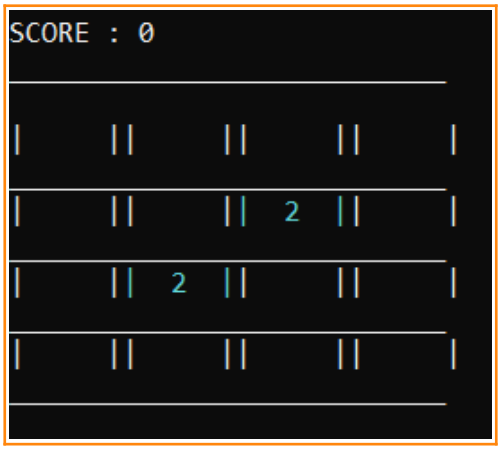
1. Objectif

Au cours de ce mini projet d'informatique, nous avons conçu un programme en C qui implémente le jeu 2048. C'est un projet pédagogique qui a pour but de mettre en application les connaissances que nous avons apprises lors du cours d'algorithmique. Ce projet fera intervenir des notions telles que les tableaux dynamiques, et plus généralement l'ensemble des bases de la programmation en C apprise en début d'année, mais également de consolider notre façon de travailler en groupe.

2. 2048, c'est quoi ?

Le jeu 2048 est un jeu de réflexion, qu'on pourrait même qualifier de casse-tête. En effet, le but du jeu est d'obtenir à partir d'une grille de 4 par 4, une case portant la valeur 2048. Mais alors comment faire ?

Et bien les règles sont simples, au début d'une partie un plateau est initialisé, totalement vide à l'exception de 2 cases portant la valeur « 2 ».



Screenshot de notre plateau de jeu 2048

Le joueur effectue alors un déplacement dans la direction qu'il souhaite : HAUT, BAS, GAUCHE, DROITE. (Respectivement « z », « b », « q », « d » avant de valider notre choix en appuyant sur la touche « entrer »).

Ainsi, de part cette action, l'ensemble des cases du tableau vont se tasser dans la direction saisie précédemment. Alors, si deux cases portant la même valeur se tassent selon la même direction, celles-ci fusionnent et la nouvelle de cette case s'ajoutent au score précédent.

Exemple : SCORE = 2 case portant la valeur 4 sur le plateau SCORE =8

3. Cahier des charges

Il est demandé de développer un programme qui permet à 1 joueur ou 2 de faire une partie selon les règles énoncés ci-dessus.

Pour cela, le travail peut être séparé en plusieurs parties :

- Au début d'une partie, le plateau initial est crée.
- Affichage d’un menu permettant de sélectionner notre type de plateau, ou si l’on souhaite charger une partie précédemment sauvegardée.
- Les utilisateurs sont invités à saisir leur coup.
- Le programme joue le rôle d'un arbitre et vérifie que les déplacement réalisés s’effectue selon les règles du jeu, et fusion des cases selon celles-ci.
- Affichage du plateau après coup, et apparition d’une nouvelle case portant la valeur « 2 », ou la valeur « 4 ».
- Le programme termine la partie lorsque le joueur à atteint la case 2048, ou lorsque le plateau est bloqué.
- La partie est sauvegardée.

4. Structure du programme

Dans cette partie, nous allons nous intéressés à comment les différentes fonctions réagissent-elles entre elles, en clair comment avons nous conçus le programme de notre jeu de 2048.

4.1. La conception et l'initialisation du plateau.

D'abord concernant le plateau, après la création d'un plateau de dimension 4x4, on rempli 2 cases de ce tableau pendant la valeur 2, à l'aide de la fonction *rand()* :

On initialise un tableau de dimension 4 x 4, et on assigne à l'ensemble de ses cases la valeur 0. On cherche alors à initialiser la situation de départ, c'est-à-dire à remplir 2 cases de la valeur « 2 » :

Le code

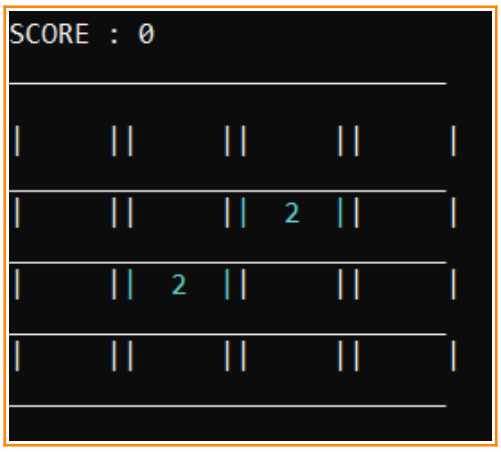
```
int nbgen=rand()%3+1;
   int nbgen2=rand()%3+1;

   int nbgen3=rand()%3+1;
   int nbgen4=rand()%3+1;

   plateau[nbgen][nbgen2]=2;
   plateau[nbgen3][nbgen4]=2;
```

On génère aléatoirement 4 entiers entre 1 et 3.

On assigne une valeur au case, avec les coordonnées générées.

Afin d'afficher le plateau, on réalisera 2 boucles for. La première balayera les lignes de notre tableau, la seconde boucle, qui elle sera contenue dans la première, balayera les colonnes de notre tableau.

Ainsi, en fonction de la valeur de la case auquel on se situe un affichage particulier se fera. C'est-à-dire que si la case du tableau pointée par les boucles contient la valeur « 2 », alors une case portant la valeur « 2 ». Voici le résultat de notre initialisation de jeu :



Screenshot de notre plateau de jeu 2048

4.2. Le déplacement des cases

Basiquement, un joueur peut tasser ses cases dans la direction qu'il souhaite ; le procédure *entasser_(DIRECTION)*, permet le déplacement, avant de réaliser la fusion de certaines cases selon les règles du jeu, par la procédure *deplace_(DIRECTION)*. Après ce déplacement suivi de la fusion correspondante, nous devons faire apparaître une case de valeur « 2 » ou « 4 », ce qui sera permis par la procédure *spawn()*.

Nous allons nous pencher sur un déplacement vers le bas.

```
void entasser_b(int plateau[4][4])
{
	int i,j,z;
	for (i=0;i<4;i++)
	{
		z=3;
		for(j=3;j>=0;j--)

if (plateau[i][j]!=0)
{
	plateau[i][z]=plateau[i][j];
	if(z>j)
	{plateau[i][j]=0;}
		z--;
		}
	}
}
```

On place une boucle pour parcourir les colonnes , on initialise une variable z à 3 qui prend donc la même valeur que j pour la dernière ligne(donc la première boucle du *for*).
Si la case n'est pas vide, sa valeur sera attribué à la case d'en dessous et z sera décrémenté de 1. La case initial sera donc remis à 0.

```
void deplace_b(int plateau[4][4])
{
 int i,j;
 for(i=0;i<4;i++)
   for(j=3;j>0;j--)
          if(plateau[i][j]==plateau[i][j-1])
                                {
  plateau[i][j]=plateau[i][j]+plateau[i][j-1];

             score=score+2*plateau[i][j-1];
    plateau[i][j-1]=0;
                                }
}
```

On crée une double boucle qui va parcourir les colonnes de manière croissante et les lignes de manière décroissante. Si deux cases ont la même valeurs pendant notre déplacement,on additionne les valeurs des 2 cases et on met à jour le score en lui rajoutant la valeur de la case. Pour finir, on vide la case qui a été impliquée dans l'addition précédente

Une fois l'entassement et la fusion réalisée, on cherche à faire apparaître une case portant la valeur « 2 » ou « 4 ».

On génère 2 entiers, qui représente les coordonnées de la case que l'on va créer. On test évidemment si la case que l'on va créer n'est pas occupée. Si elle l'est, alors on crée de nouvelles coordonnées.

On génère ici un entier entre 1 et 2.
En fonction de la valeur de ce dernier, la valeur de la case que l'on va créer sera de « 2 » ou « 4 »

```
void spawn(int plateau[4][4]){

  srand(time(NULL));

  int nbgens;
  int nbgene;

  int gener;
  do {
    nbgens=rand()%3+1;
    nbgene=rand()%3+1;
  }while(plateau[nbgens][nbgene]!=0);

  int alea;
          alea=rand()%2+1;
  int val;
  if(alea==1){
                    val=4;
  }
  else{
                    val=2;
 }
  plateau[nbgens][nbgene]=val;
```

4.3 La partie est terminée !

Il existe plusieurs fin possibles du jeu 2048. La première fin possible, est celle si le joueur arrive à obtenir une case portant la valeur « 2048 » selon les règles précédentes.

La seconde fin, est celle selon laquelle l'ensemble du plateau est bloqué. C'est à dire, si toutes les cases du plateau porte une valeur, mais que tous les déplacements sont impossibles.

Ces fins de jeu sont permises par les fonctions / procédure suivantes :

```
void fin(int plateau[4][4]){

 int hauteur;
 int largeur;

 for(largeur=0;largeur<4;largeur++){
  for(hauteur=0;hauteur<4;hauteur++){

    if(plateau[largeur][hauteur]==2048){

     printf("Gagner !");

      exit(0);
,,,
}
```

Une victoire :

Nous avons réaliser deux boucles, qui vont ainsi balayer le plateau de jeu, à la recherche d'une quelconque case portant la valeur « 2048 ». Si cette valeur est présente, le jeu s'arrête immédiatement, et vous avez alors GAGNE !

Une défaite :

Nous avons découper le processus de vérification de la condition de défaite en 2 parties. Tout d'abord, nous allons vérifier si le plateau est entièrement rempli, ou non.

Le plateau est entièrement rempli, on vérifiera alors s'il reste des déplacements possibles sur le plateau de jeu.

Cette vérification est permis par la procédure *fin_2()*, expliquée à la page suivante.

On constatera, que nous ne ajouterons pas de valeur au plateau de jeu, à cette étape-ci.

Si il reste un déplacement possible dans notre plateau, alors le joueur est invité à saisir un déplacement. Après celui-ci ; on vérifiera à nouveau si le plateau est entièrement rempli ou non :

Le plateau n'est pas entièrement rempli, on peut ajouter une case

Le plateau est toujours rempli, on l'affiche puis on passe au tour suivant.

Le plateau n'est pas entièrement rempli, par conséquent, le joueur peut jouer de façon normal, c'est-à-dire effectuer un déplacement avant de voir une nouvelle case apparaître sur son plateau, et de vérifier si il n'a pas gagner sa partie.

```
for(largeur=0;largeur<4;largeur++){
   for(hauteur=0;hauteur<4;hauteur++){
    if(plateau[largeur][hauteur]!=0){
     cpt++;
    }
   }
  }

  if(cpt==16){

   fin_2(plateau);

   SaisieDirection(plateau);

   fin(plateau);



   for(largeur=0;largeur<4;largeur++){
            for(hauteur=0;hauteur<4;hauteur++){
             if(plateau[largeur][hauteur]!=0){
              cpt++;
             }
              if(cpt!=16){
               spawn(plateau);
               affiche_ligne(plateau);
              }
              else{
               affiche_ligne(plateau);
              }



   }
   else{
    SaisieDirection(plateau);
    fin(plateau);
    spawn(plateau);
    affiche_ligne(plateau);
   }
```

La procédure *fin_2()*, nous permet de vérifier si il reste des déplacements possibles à effectuer, si l'on se réfère aux règles du jeu de 2048.

```
void fin_2(int plateau[4][4]){

 int i,j;

 int ct;

 int a,b;


       for(i=0;i<4;i++){
       for(j=0;j<4;j++){

          a=i;
          b=j;

         if(ct==16){
           printf("Perdu!\n");
           exit(0);
         }


      if(i==0 && j==0){
 if( (plateau[a][b]!=plateau[a+1][b]) && (plateau[a][b]!=plateau[a]
[b+1]) ){
           ct++;
         }

         }
```

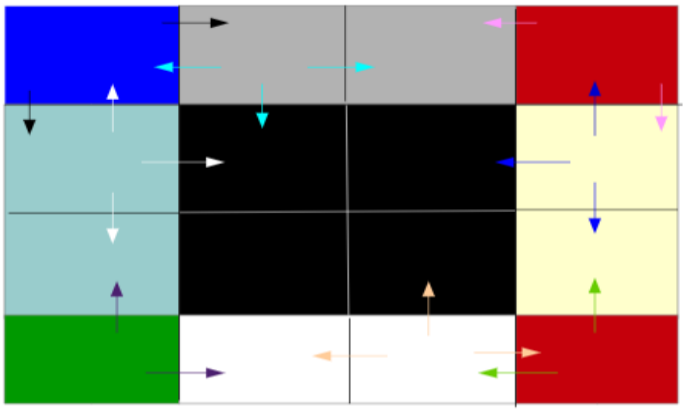
Ces deux boucles *for* balayeront le plateau de jeu :

Si le nombre cases bloqués (*compteur ct*) est égal à 16, alors cela signifie qu'aucun déplacement n'est permis, la partie est PERDU

En fonction de la case pointée, on vérifie si les déplacement autorisé sont effectuable.

On vérifiera que pour la case pointée, les cases ou l'on peut se déplacer sont égales à cette dernière. Si ce n'est pas le cas, on incrémentera le compteur ct.

On se référera au schéma suivant :



4.4 Sauvegardons notre partie / chargeons notre partie !

Pour sauvegarde notre partie, nous avons crée une procédure qui nous permet de sauvegarder l'ensemble des nombres contenu dans notre tableau (donc dans toutes les cases du plateau) au sein d'un fichier texte.

L'idée est de pouvoir stocker ces valeurs en donnant en paramètre de cette procédure notre tableau afin de pouvoir sauvegarder la progression en cours du joueur.

Après avoir définie nos variables, on ouvre notre fichier appeler ici «test.txt» grâce a la fonction **fopen** suivi de l'argument 'a' afin de pouvoir grâce a la fonction **fprintf** en écrasant le contenu actuel du fichier, ceci afin d'éliminer les données sauvegardées d'une éventuelle partie précédemment sauvegardée afin de les remplacer par les nouvelles.

A l'aide d'une boucle qui parcours l'ensemble des cases de notre plateau, on écrit dans notre fichier texte l'ensemble des valeurs contenu à l'intérieur, en sautant une ligne dans le document texte entre chaque valeur récupérer afin de faciliter la lecture du fichier lors du chargement.

```
void sauvegardepartie(int plateau[4][4]){

  FILE* ecriture;
  char* nomfichier= "test.txt";

  for (int i = 0; i < 4; ++i)
  {
    for (int j = 0; j < 4; ++j)
    {
      ecriture= fopen(nomfichier,"a");
      fprintf(ecriture,"%d\n",plateau[i][j]);
      fclose(ecriture);
...}
```

Une fois la partie sauvegarde, il suffit d'appuyer sur la touche «1» comme le montre le screenshot suivant afin de démarrer la partie à partir de l'instant précédemment sauvegardé.

```
int chargerpartie(int plateau[4][4]) {

  FILE* lecture;
  char* nomfichier= "test.txt";
  char ligne[MAXCARAC];
  lecture=fopen(nomfichier,"r");
  int i=0;
  int j=0;
```

Afin de charger la partie, il suffit donc de procéder de la même manière en sens inverse, c'est-à dire que nous allons ouvrir notre fichier et le lire avec la fonction **fgets** qui va nous permettre de lire notre fichier ligne par ligne (d'où l'intérêt de sauter une ligne entre chaque valeurs lors de la sauvegarde de notre fichier)

```
while(fgets(ligne,MAXCARAC,lecture)!= NULL){
 // boucle pour lire toute les lignes d'un code

    plateau[i][j] = atoi(ligne);

    if(j==3){
      j=0;
      i=i+1;
    }

    else{
      j=j+1;
    }
 }

 fclose(lecture);
 return plateau[i][j];
}
```

Nous avons donc crée une boucle qui permet de récupérer la valeur de chaque ligne tant que cette dernière n'est pas nul. Les éléments présents dans notre document texte sont des caractères, donc il faut utiliser la fonction **atoi** juste avant de rentrer nos valeurs dans notre plateau afin de convertir ces caractères en entier.

Pour finir, il suffit de fermer notre fichier après avoir lu toutes nos valeurs grâce à la fonction **fclose**, et de retourner le plateau remplis avec les valeurs récupérer.

5. Réalisation des tâches

Dans le tableau ci-dessous nous avons répertorié les tâches réalisés par chaque membre de notre groupe :

Membre	Yannis	Mohamed	Tâche réalisée ?
Plateau	✔		✔
Sauvegarde		✔	✔
Fin	✔		✔
Déplacement	✔	✔	✔
Programme	✔	✔	✔
Spawn		✔	✔
Rapport	✔	✔	✔

6. Difficultés rencontrées

Durant notre travail, nous avons rencontré différents problèmes qui ont ralenti l'avancement souhaité de notre projet.

En effet, nous avons rencontrés quelques bugs récurrent concernant les déplacements selon certaines direction, que nous avons pu corriger et améliorer à l'aide de notre professeur.

Dans l'ensemble du projet, la structure du programme du jeu 2048 se rapprochait fortement de celle du programme du jeu de Traverse.

7. Conclusion

Afin de conclure sur notre projet d'informatique, nous pouvons affirmer qu'il est très intéressant car il nous a permis d'appliquer à la fois nos connaissances en algorithmique et en C apprise tout au long de cette année en classe, et ainsi de pouvoir les appliquer à un cas concret. Malgré certaines difficultés le projet est très formateur et instructif pour nous, car il nous a permis d'apprendre à s'organiser dans un travail de groupe, et de confronter chacune de nos idées. Il nous a été bénéfique car il nous a aussi permis de consolider nos acquis, mais aussi de réfléchir aux diverses stratégies dans le but de résoudre un problème.