Rapport de projet d'études :

Projet C

Membres du groupe :

Rym ABED
Rime BENZOUINE
Nouhaila EZZAHR

Année d'études :

ING1

Session :

2021/2022

Tuteur :

Mr. SABLON

Tables des matières :

Introduction :

Notre projet a pour objectif de programmer le Morpion en langage C. Et nous avons choisi Code::Blocks comme environnement de développement. La programmation s'est faite en respectant les règles officielles du jeu et en intégrant les fonctionnalités que le cahier des charges nous impose. En effet, notre jeu permet au joueur de :

- Choisir la taille de la grille du jeu
- Choisir un mode de jeu (jouer contre un deuxième joueur ou une IA)
- Personnaliser le pion (forme, couleur du pion du joueur 1 et/ou 2)
- Choisir l'intelligence de l'ordinateur (qui est proportionnelle à la difficulté)
- Revisualiser une partie
- Sauvegarder/Charger une partie
- Consulter les statistiques de chaque joueur

Ce projet a été suivi par Mr SABLON.

Organisation du projet :

Pour accomplir ce projet, nous nous sommes partagés les fonctionnalités du jeu et nous nous sommes constamment rencontrés à la bibliothèque ou chez Rime pour mettre en commun nos codes.

Tout d'abord, et comme la programmation en langage C n'est pas maitrisée par tous, une formation s'est imposée afin de garantir une bonne maitrise du langage et un bon avancement par la suite.

Dans un premier temps, nous avons programmé le jeu et la grille du jeu.

Le code du jeu permet au joueur de choisir un numéro de ligne ensuite un numéro de colonne pour enfin placer le pion à l'endroit que le joueur souhaite.

Quant à celui de la grille, il consiste à déposer de manière graphique les caractères ' | ' et '-' afin de créer les différentes cases.

Dans un deuxième temps, nous nous sommes intéressés aux fonctionnalités exigées par le cahier des charges.

Nous avons commencé par le choix du mode de jeu en programmant d'abord le mode joueur contre joueur avec la mise en place d'un code permettant de switcher d'un joueur à l'autre. Par la suite, nous avons programmé le mode joueur contre IA en intégrant la possibilité de choisir un niveau de difficulté (de facile à difficile) ce qui rend le jeu plus amusant.

Ensuite, nous avons écrit les fonctions permettant la personnalisation du pion (couleur et forme) et d'autres permettant à chacun des deux joueurs de personnaliser son pion.

Dans un troisième temps, nous avons mis en place un menu qui regroupe toutes les fonctionnalités numérotées. Pour choisir une fonctionnalité, il suffit de taper son numéro.

Dans un quatrième et dernier temps, nous avons mis en place le code des statistiques du jeu et abordé la question sur comment sauvegarder une partie du jeu ainsi que la re-visualisation de celle-ci. Pour la sauvegarde, nous avons opté pour le stockage des positions du pion de chacun des deux joueurs dans un fichier nommé « Save ». Ainsi, pour charger et/ou revisualiser une partie, il suffit de le lire.

Répartition des tâches :

Nous nous sommes répartis les tâches de la manière suivante :

- **Rym :** La conception du jeu Morpion ainsi que le programme qui permet de jouer à deux personnes sur un même ordinateur et la grille.
- **Rime :** La conception du menu en le reliant aux fonctions du jeu et aux différentes options du jeu (modifié par **Rym** afin de permettre une meilleure lecture et une meilleure compréhension).
- **Nouhaila :** La conception du programme qui permet au joueur de jouer contre une intelligence artificielle (adapté par **Rym** aux fonctions définies au préalable dans le mode à deux joueurs).
- **Rime :** La conception des options graphiques du jeu (forme et couleur du pion).
- **Nouhaila :** L'ajout du choix du niveau de difficulté.
- **Rime :** L'ajout de la fonctionnalité de sauvegarde de la partie dans un fichier.
- **Nouhaila :** L'ajout de la fonctionnalité de chargement d'une partie.
- **Rime :** La re-visualisation d'une partie déjà jouée (adapté par **Rym** afin de permettre l'affichage correct de la grille).
- **Nouhaila :** L'ajout des statistiques du jeu.
- **Rym :** Rédaction du rapport
- **Nouhaila et Rime :** Rédaction du pseudo-code

Pseudo-code:

1)GameMenu() :

```
begin
        while out!=0 do
                afficher le menu
                principal ;get choice ;
                switch(choice)
                        1) SwitchPlayer() ;
                        2) RowsNumber() ;
                        3) PawnChoice() ;
                        4) GameMode() ;
                        5) GameDifficulty() ;
                        6) Statistics() ;
                        7) ReviewPart();
                        8) LoadPart() ;
                        9) Rules();
                        10)     exit(0) ;
        end while
end
```

2)Grid():

```
begin
        for h<rownumber do
                afficher « %d » ;
        end for
        for h<4* rownumber do
                afficher «-» ;
        end for
        for i< rownumber do
                if i>=9
                        afficher le numéro de ligne ;
                else
                        afficher aussi le numéro de ligne mais différemment (pour les
                dizaines) ;
                endif
```

```
            for j< rownumber do
                afficher « | » ;
                switch(grd[i][j])
                    0) afficher « »
                    1) afficher le pion du joueur 1 de la couleur correspondante ;
                    2) afficher le pion du joueur 2 de la couleur correspondante ;
            end for
            afficher la dernière limite verticale ;
            for h<4* rownumber do
                    afficher la dernière limite horizontale ;
            end for
        endfor

end
```

3)SwitchPlayer():

```
begin
        memset(0,Grd) ;
        Grid() ;
        Sfile=fopen(fichier_sauvegarde)
        ;fclose(Sfile) ;
        while ((!WinGame()) and
                (!EqualGame())) do
                Game(Grd(player) ;
                if (player==1) then
                        player=2 ;
                  else
                          player=1;
                  endif
           endwhile
           return 0;

end
```

4)RowsNumber():

```
begin
        while out=!0 do
                afficher ce
                menu ;get
                choice ;
                if 2> rownumber > 11 then
                        out=0 ;
                else
                         afficher « nombre
 incorrect » ;
                                out=1 ;
                  endif
          return rownumber ;

 end
```

5)PawnChoice():

```
begin
       while out!=0 do
                afficher
                ce menu ;
                get
                choice ;
                switch(ch
                oice)
                 1) ChoicePawnPlayer1(pawnShape1, pawnColor1) ;
                 2) ChoicePawnPlayer2(pawnShape2, pawnColor2) ;
                 3) out!=0 ;
end
while
end
```

6)Color():

```
begin
        SetConsoleTextAttribute(H,couleurdefond*16+couleurdutexte) ;
end
```

7)PawnColor():

```
begin
        while out!=0 do
                afficher ce menu avec les couleurs écrites dans la couleur
                correspondante ;get Pion ;
                switch(TicTacToeChoice)
                        (pour les 15 couleurs reproduire le schéma du case)
                        1) afficher la couleur choisie ;
                          l'attribuer au joueur sélectionné dans le menu ;
                        …
                        15) afficher la couleur choisie ;
                           l'attribuer au joueur sélectionné dans le menu ;

        endwhile
         return TicTacToeChoice ;
end
```

8)PawnShape():

```
begin
        while out!=0 do
                afficher ce
                menu ;get
                choice ;
                switch(TicTacT
                oeChoice)
                        1) Forme 'O' pour le joueur qui choisit ;
                        2) Forme 'X' pour le joueur qui choisit ;
                        3) Forme '+' pour le joueur qui choisit ;
                        4) Forme '$' pour le joueur qui choisit ;
                        5) Forme '@' pour le joueur qui choisit ;
          endwhile
           attribut la forme au joueur qui choisit ;
end
```

9)ChoicePawnPlayer1():

```
begin
        while out!=0 do
                afficher ce
                menu ;
                switch(choice)
                        1) PawnShape ();
                        2) PawnColor();
                        3) out=0;

        endwhile

end
```

10)ChoicePawnPlayer2():

```
begin
        while out!=0 do
                afficher ce
                menu ;
                switch(choice)
                        1) PawnShape ();
                        2) PawnColor();
                        3) out=0;

        endwhile

end
```

11)GameMode():

```
begin
        while out!=0 do
                afficher ce
                menu ;get
                choice ;
                switch(TicTacT
                oeChoice)
          1) contre un joueur ;
```

```
        2) contre ia ;
        3) out;

      endwhile
      change l'affichage du menu principal ;
end
```

12)GameDifficulty():

```
begin
      while out!=0 do
            afficher ce
            menu ;get
            choice ;
            switch(choice)
                  1)difficulté = facile ;
                        change l'affichage menu principal ;
                  2)difficulté = moyen ;
                        change l'affichage menu principal ;
                  3)difficulté = difficile ;
                        change l'affichage menu principal ;
                  4)out;
       endwhile
      attribut la difficulté ;
 end
```

13)Statistics():

```
begin
      while out!=0 do
            afficher les
      statistiques ;
            get choice ;
             switch(choice)
                  1)out;
end
```

14)ReviewPart():

```
begin
        chargement grille vide ;
        ouverture fichier
        sauvegarde ;retour
        début du fichier ;
        if (Freview!=NULL) then
                while(posX!=EOF and posY!=EOF)
                        dowhile(correct!=TRUE) do
                                afficher le joueur ;
                                attribuer la case grd([posX][posY]) au
                                joueur ;correct=TRUE ;
                end while
                if(player==1) then
                         player=2 ;

                endif
                if(player==2)
                then
                         player=1
                ;
                endif
        else
                 prevenir qu'il n'y a pas de partie à charger ;
         endif
end
```

15)LoadPart():

```
begin
        initialisation du plateau avec
        des 0 ;if Freview!=NULL then
                while (posX and posY du fichier) !=
                        EOF
                        while correct!=TRUE
                                afficher joueur ;
                                grd[posX-1][posY-1]=player ;
                                correct=TRUE;
```

```
                    end while
                    Grid()
                    if player==1
                    then
                        player=2 ;

                    else
                        player=1;

                    endif

                endwhile

            else
              Annoncer: Pas de partie à
            charger!
            endif
            while((!WinGame() and Vreview==TRUE) or (!EqualGame() and
                Vreview==TRUE)) doif WinGame() or EqualGame() then

                    Vreview=FALS
                    E;
                    GameMenu() ;
                    return ;
                if Vreview==TRUE then
                    Game(Grd,player
                ); if( player==1 )
                    player=2;

                 else
                    player=1;

                endif


            endwhile
            return;
end
```

16)Rules():

```
begin
while out!=0 do
        afficher règles ;
        get choice ;
        switch(choice)
                1)out=0 ;
 end
```

17)Data():

```
begin
   while ((x>1 and y==0) and (x<RowNumber))
         doafficher le joueur qui joue ;
        get x ;
        y=1 ;
  end while
  y=1 ;
  return x ;

end
```

18)BackUp():

```
begin
        FSauv=fopen(fichier de
        sauvegarde) ;if Fsauv == NULL
        then
                afficher que le fichier est impossible à ouvrir ;
        else
                 ecrire les valeurs des positions x et y dans le fichier ;
        endif
        fclose(FSauv) ;


end
```

19)random():

```
begin
        srand(time(NULL));
        randomNumber =(rand()%RowNumber) ;
                return randomNumber ;
end
```

20)Game():

```
begin
        for i<=RowNumber do
                créer un tableau avec le numéro de chaque ligne/colonne ;
        end for
        while correct!=TRUE do
                if (player==1) or (player==2 and
                        robot==1) thenafficher joueur ;
                        posX = Data() ;

                        posY = Data() ;

                        if posX!=a[posX] or posY!=a[posY]

                                thenGameMenu() ;
                                correct=TRUE
                                ;return ;
                        if grd[posX-1][posY-1]>0

                                        afficher case
                                occupée ;
                        if grd[posX-1][posY-1]==0

                                        if player==1
                                        ajoute 1 au nombre de coups du
                                joueur ;if(player==2)
                                        ajoute 1 au nombre de coups du joueur ;
                                end if
                                grd[posX-1][posX-1]=joueur ;
                                correct=TRUE;

                                BackUp(Fbackup, posX, posY);
                        end if
```

```
            if player==2 and
                    robot==2 then

                        while
                    Test!=FALSE do
                        for i<RowNumber do
                            for j<RowNumber do
                                        effectuer tous les tests pour compléter des
                                                    alignements ;
                                end for
                            end for

                        for i<RowNumber do
                            for j<RowNumber do
                                        effectuer tous les tests pour bloquer des
                                                    alignements ;




                                    endfor
                                endfor
                    endwhile
                    while secondtest==TRUE do
                            selectionner une case au hasard et vérifier si elle est
     vide avant de la
                            compléter;
                    end while
                endif
        endwhile
        Grid();
end
```

21)WinGame():

```
begin
        for i<=RowNumber do  (test lignes)
                for j<RowNumber-2 do
                        if (grd[i][j]>0 and grd[i][j]==grd[i][j+1]==grd[i][j+2]) then
                        if player==1 then
```

```
                                afficher victoire joueur
                                ; ajouter 1 point au
                                gagnant ;return TRUE ;
                        end if
                        if player==2 then
                                afficher victoire joueur
                                ; ajouter 1 point au
                                gagnant ;return TRUE ;
                        endif
                endif

        endfor
    endfor
    for i<RowNumber-2 do   (test des colonnes)
        for j<RowNumber-2 do
            if (grd[i][j]>0 and grd[i][j]==grd[i+1][j]==grd[i+2][j]) thenif

                player==1 then
                        afficher victoire joueur
                        ; ajouter 1 point au
                        gagnant ;return TRUE ;

                 endif
                if player==2 then
                        afficher victoire joueur
                        ; ajouter 1 point au
                        gagnant ;return TRUE ;
                endif
            endif
         endfor
    endfor
    (et la meme chose pour les diagonales aussi )
     return FALSE;
end
```

22)EqualGame():

```
begin
    for i<RowNumber do
        for j<RowNumber do
            if grd[i][j]==0 then
```

```
                        return FALSE ;
                    endif
        endfor
        afficher « Egalité » ;
        return TRUE ;
end
```

Connaissances acquises :

Nous avons acquis de nombreuses connaissances suite à la réalisation de ce projet.

Tout d'abord, nous avons appris à programmer en langage C qui est considéré comme un langage de bas niveau difficile à maitriser. Ce qui nous facilitera par la suite l'apprentissage de d'autres langages de programmation et favorise un CV.

Nous avons globalement appris à programmer, à changer la couleur du texte, à ouvrir un fichier puis écrire et sauvegarder dedans, à approfondir nos connaissances sur les fonctions propres au langage C.

La réalisation d'un projet de programmation à 3 nous a appris à travailler en équipe, s'entraider, communiquer, se comprendre et s'adapter les uns aux autres afin de maintenir une bonne ambiance de travail et ainsi un bon avancement du projet.

De plus, nous avons appris à se répartir les tâches en fonction des compétences et des disponibilités de chaque membre. En effet, nous avons repéré les points forts de chacun ainsi que ses disponibilités. En se basant sur ces deux faits, nous nous sommes réparties les tâches.

Difficultés rencontrées :

Nous avons rencontré de nombreuses difficultés en réalisant ce projet.
Tout d'abord, étant dans des groupes TP différents, nous avons eu du mal à nous réunir. Ce qui nous a parfois retardé par rapport au planning initial.

De plus, l'ajout de l'option chargement/re-visualisation d'une partie nous a pris beaucoup de temps car on n'avait pas d'idée sur ce qu'on doit sauvegarder au juste. Finalement, nous avons opté pour une sauvegarde des positions du pion de chaque joueur dans un fichier. Nous avons donc dû étudier les différentes fonctions liées aux fichiers telles que la fonction fprintf, fscanf, fopen et fseek.

Nous avons eu du mal à développer la fonctionnalité qui redemande au joueur de rentrer un autre numéro de ligne et de colonne lorsque celui-ci se trompe. Car cette fonctionnalité nous a créée un problème dans le programme qui permet de quitter une partie en plein milieu. Afin que ce soit plus simple nous avons choisi un programme qui interrompt la partie si le joueur rentre 0 et lui donne infiniment de chance de rentrer un nouveau numéro de ligne/colonne lorsqu'il rentre un nombre supérieur au nombre de ligne de la grille.

Certains codes nous ont pris plus de temps que nous avons estimé lors de la répartition des tâches, mais en s'entraidant nous avons réussi à rattraper ce retard.

La révision pour les partiels/les examens nous a également retardé un peu pour faire les dernières modifications du projet.

Améliorations et évolutions possibles :

Comme tout programme, notre programme n'est pas parfait et nécessite quelques améliorations.

En effet, nous pourrons concevoir une grille de plus que 10lignes. Même si ce n'est pas difficile à faire, cette fonctionnalité entraîne de nombreuses erreurs sur notre code. Nous avons donc préféré rester sur des fonctionnalités limitées et un code fonctionnel.

Nous pourrons aussi augmenter la performance de l'ordinateur en utilisant l'algorithme MinMax qui permet à l'ordinateur de jouer toute la partie avec toutes les prévisions possibles pour ensuite choisir le meilleur chemin pour gagner.

Nous pourrons également empêcher les deux joueurs de choisir la même forme et couleur du pion afin d'éviter les confusions.

Nous pourrons fixer un nombre d'essai (chance) à ne pas dépasser lorsqu'on mentionne une ligne/colonne à l'extérieur de la grille.

Nous pourrons ajouter l'option de choisir qui commence en premier entre le joueur et l'ordinateur.

Conclusion :

Notre projet a été réalisé dans le respect des règles officielles du jeu ainsi que le cahier des charges.

Les difficultés auxquelles nous avons fait face nous ont beaucoup appris en termes de maitrise des fonctionnalités du langage C et les techniques de codage.

Cependant, notre projet peut être améliorer et perfectionner afin de rendre le jeu plus amusant.