```
#include "grille.h"
#include "joueur.h"
#include "IntArt.h"
#include "partie.h"
#include <cstdlib>

using namespace std;

////////////////// FONCTIONS HORS-CLASSE USUELLE ///////////////////////

bool string_is_int(string coup)     //test si un string est un nombre
{
        if(atoi(coup.c_str()))      //si c'est un int
                return true;
        return false;
}

int string_to_int(string coup)      //transforme les string en int
{
        int i=atoi(coup.c_str());       //transformation en int
        return i;
}

////////////////// CONSTRUCTEUR ET DESTRUCTEUR ////////////////////////////

Partie::Partie()
{
        nb_joueur=0;
        nb_IA=0;
        difficulte=0;
        limitateur_de_changement=0;
        J=NULL;
        IA=NULL;
}

Partie::~Partie()
{
        delete[] J;         //on supprime le tableau de joueur
        delete[] IA;        //on supprime le tableau d'IA
}

////////////////// MENUS ET SOUS-MENUS ////////////////////////////////////

void Partie::menu_principal()
{
        system("clear");
        string navigation;      //permet de selectionner un menu dans le menu principal
        do
        {
                system("clear");
                cout<<endl<<endl<<" **************X O  PUISSANCEx INVERSE X X
                O******************"<<endl;
                cout<<" **************O X X O O X O X X X O O O X O******************"<<endl;
                cout<<" *                                                          *"<<endl;
                cout<<" *                                                          *"<<endl;
                cout << "   *\E[33;1m            MENU
                PRINCIPAL\E[m                          *"<<endl;
                cout<<" *                                                          *"<<endl;
                cout<<" *                                                          *"<<endl;
                cout<<" *   Demarrer une nouvelle partie: Tapez 1                *"<<endl;
                cout<<" *                                                          *"<<endl;
                cout<<" *   Charger une partie: Tapez 2                          *"<<endl;
                cout<<" *                                                          *"<<endl;
                cout<<" *   Acceder aux règles du jeu: Tapez 3                   *"<<endl;
                cout<<" *                                                          *"<<endl;
                cout<<" *   Quitter: Tapez 4                                     *"<<endl;
                cout<<" *                                                          *"<<endl;
                cout<<" *                                                          *"<<endl;
                cout<<" *                                  copyright©  DELMAS Philippe  *"<<endl;
                cout<<" ************************************************************"<<endl;
                cin>>navigation;    //on selectionne l'entrée desirée
```

```
                system("clear");
                if(navigation=="1")     //sinouvelle
                {
                        joueurs_et_IA();    //premiere etape de la configuration de partie
                        (config des joueur)
                        system("clear");
                        G.saisie();     //deuxieme etape de la configuration de partie
                        (config de la grille)
                        for(int i=0; i<nb_joueur; i++)
                                G.recupere_jeton_joueur(i+1, jeton_joueur(i+1)); //on envoi
                                les jetons des joueur dans la grille
                        for(int i=nb_joueur-1; i>=nb_joueur-nb_IA; i--)
                        {
                                int num_ia=i-nb_joueur+nb_IA;
                                IA[num_ia].recupere_grille(G);
                                IA[num_ia].recupere_jeton();
                        }
                        do
                        {
                                system("clear");
                                cout << "\E[35;1m           VALIDATION\E[m"<<endl<<endl<<endl;
                                cout<<"     Les spécification de cette partie ont bien été
                                mémorisé:"<<endl<<endl;
                                cout<<endl<<"       Je veux les changer (retour au menu
                                principal): appuyer sur 0"<<endl;
                                cout<<"     Elles me conviennent (COMMENCER LA PARTIE):
                                appuyer sur 1"<<endl;
                                cin>>navigation;    //pour permettre au joueur de modifier
                                les données si il a fait une erreur
                        }
                        while(navigation!="1"&&navigation!="0");
                        if(navigation=="1")
                                tour_par_tour(1);   //on commence la partie avec le joueur 1
                        else if(navigation=="0")
                                menu_principal();   //on retourne au menu principal
                }
                else if(navigation=="2")    //pour charger une partie
                {
                        string nom;     //nom de la partie a charger
                        string str;     //nom du fichier reel (avec extension et chemin pour
                        y acceder)
                        cout<<"quel est le nom du fichier a charger (SANS le chemin et SANS
                        l'extansion .txt):"<<endl;
                        cin>>nom;
                        str="base_de_donnee/"+nom+".txt";   //concaténation de string
                        int joueur_qui_joue=G.charger(str);     //le joueur qui a sauver est
                        celui qui doit commencer(son index est retournée par la fonction
                        charger)
                        if(joueur_qui_joue==0)
                            menu_principal();
                        nb_joueur=G.renvoi_nb_joueur();     //on recupere le nombre de joueur
                        J=new joueur[nb_joueur];
                        charger_jeton_joueur();     // on recupere les jetons de joueur
                        tour_par_tour(joueur_qui_joue);     //on retourne a la partie, le
                        joueur qui commence est celui qui a sauvegarder)
                }
                else if(navigation=="3")    //pour voir les regle du jeu
                        regle_du_jeu();
                else if(navigation=="4")    //pour quitter le jeu
                {
                        system("exit");
                }
        }
        while(navigation<"1" || navigation>"4");
}

void Partie::joueurs_et_IA()
{
        string nb;      //nombre de joueur
        do
        {
```

```
                system("clear");
                cout << "\E[31;1m           REGLES DE CETTE PARTIE\E[m"<<endl<<endl<<endl;
                cout<<"     saisir le nombre de joueur(2, 3 ou 4):  ";
                cin>>nb;
                nb_joueur=string_to_int(nb);
                if(nb_joueur>=2&&nb_joueur<=4)
                {
                        cout<<endl<<"       saisir le nombre d'IA (entre 0 et
                        "<<nb_joueur-1<<"):   ";
                        cin>>nb;
                        nb_IA=string_to_int(nb);
                        if(nb_IA>0&&nb_IA<=nb_joueur-1)
                        {
                                cout<<endl<<"       saisir le niveau de difficulte :"<<endl;
                                cout<<"         1: Coups au hasard"<<endl;
                                cout<<"         2: Tres facile"<<endl;
                                cout<<"         3: Facile"<<endl;
                                cout<<"         4: Moyennement facile"<<endl;
                                cout<<"         5: Intermediaire"<<endl;
                                cout<<"         6: Moyenement difficile"<<endl;
                                cout<<"         7: Difficile (deconseillé pour les grandes
                                grilles)"<<endl;
                                cout<<"         8: Tres difficile (deconseillé pour les
                                grandes grilles)"<<endl;
                                cin>>nb;
                                difficulte=string_to_int(nb);
                        }
                        else
                            difficulte=1;
                }
        }
        while(nb_joueur<2||nb_joueur>4||nb_IA>nb_joueur-1 || nb_IA<0||difficulte>9 ||
        difficulte<1);
        G.nombre_joueur(nb_joueur);     //on envoie le nombre de joueur a la grille
        J=new joueur[nb_joueur];
        IA=new ia[nb_IA];
        for(int i=0;i<nb_joueur;i++)
                J[i].attribuer_numero(i+1);     //on attribue un index a chaque joueur
        if(nb_IA!=0)
        {
                for(int i=1; i<nb_joueur; i++)
                        J[i-1].humain_ou_ia(0);
                for(int i=nb_joueur-1; i>=nb_joueur-nb_IA; i--)
                        J[i].humain_ou_ia(1);
                for(int i=0; i<nb_IA; i++)
                        IA[i].recupere_donnee_partie(difficulte,nb_joueur,nb_IA);

        }
        system("clear");
        int couleur=33;     //couleur de depart
        for(int i=0;i<nb_joueur;i++)
        {
                J[i].selection_forme_jeton(couleur);
                J[i].attribuer_couleur(i+1);
                if(i<2)
                        couleur=couleur-2+i*5; //-2 pour avoir 31 (rouge) pour J[1] et
                        -2+i*5 ou i vaudra 1 pour avoir 36 (cyan)pourJ[2]
                else if(i==2)
                        couleur=32;
                system("clear");
        }
}

void Partie::regle_du_jeu()
{
        string nav;
        cout << "\E[31;1m           REGLE DU JEU\E[m"<<endl<<endl<<endl;
        cout << "\E[36;1m   1)Objectif\E[m"<<endl<<endl;
        cout<<"Le but du jeu est d'aligner un nombre x de jeton d'un meme joueur n
        fois."<<endl;
        cout<<"Le joueur qui aligne n fois x jetons gagne la partie. En cas d'un "<<endl;
```

```
        cout<<"remplissage complet de la grille sans les n alignements necessaires, "<<endl;
        cout<<"il y a partie nulle (PEUT IMPORTE SI UN JOUEUR POSSEDE PLUS
        D'ALIGNEMENTS"<<endl;
        cout<<"QUE LES AUTRES)."<<endl;
        cout<<"Les jetons sont placés dans une grille de longueur et largeur variable
        qui"<<endl;
        cout<<"sont definits par les joueurs lors de l'initialisation de la
        partie."<<endl<<endl;
        cout << "\E[36;1m   2)Coups possible\E[m"<<endl<<endl;
        cout<<"Les joueur dispose de plusieurs coup pour gagner:"<<endl;
        cout<<"1) Placer un pion dans la grille en selectionnant la colonne."<<endl;
        cout<<"2) Faire pivoter la grille en sens horaire ou antihoraire, ce qui fait"<<endl;
        cout<<"tomber les pion en consequence."<<endl;
        cout<<"3)Inverser ses pions avec ceux d'un autre joueur(il volent egalement
        ses"<<endl;
        cout<<"alignements mais ce coup n'est plus utilisable pendant tout un
        tour)."<<endl<<endl;
        cout <<"\E[36;1m    3)Alignements autorisés et interdits\E[m"<<endl<<endl;
        cout<<"Un alignement doit pour être comptabilisé être horizontal, vertical ou "<<endl;
        cout<<"diagonal (les deux diagonales sont autorisés)"<<endl;
        cout<<"Les jetons peuvent être utilisé plusieurs fois à condtion qu'il soient "<<endl;
        cout<<"sur deux alignement de types different (horizonal avec diagonal, vertical
        "<<endl;
        cout<<" et horizontal, diagonal gauche et diagonal droite...)"<<endl;
        cout<<"Si un jeton est sur deux alignement d'un meme types(en considerant la
        diagonale"<<endl;
        cout<<"gauche et la diagonale droite comme deux alignements differents, un seul des
        deux sera"<<endl;
        cout<<"comptabilisé"<<endl;
        cout<<"Par exemple pour un alignement 3 jetons,le cas suivant n'en considera
        qu'un:"<<endl;
        cout<<"          X   X   X   X"<<endl;
        cout<<"Pour le cas suivant il en consedera deux:"<<endl;
        cout<<"          X   X   X   X   X   X"<<endl<<endl;
        cout<<"taper quelque chose pour retourner au menu principal"<<endl;
        cin>>nav;
        menu_principal();
}

////////////////// GESTION DU TOUR PAR TOUR ////////////////////////////

void Partie::tour_par_tour(int joueur_qui_commence)
{
    limitateur_de_changement=0;     //permet de ne faire qu'un seul echange par tour
        while(G.nb_coup_possible()!=0)      //tant qu'il y a des cases vides
        {
                for(int i=joueur_qui_commence; i<=nb_joueur; i++)
                {
                    tour_du_joueur(i);  //on effectue le tour du jour i
                    if(i==nb_joueur)    //si on arrive au dernier joueur du tour
                            i=0;    //on repart au premier
                    if(G.fin()==1)      //si partie arretée
                        break;      //on sort
                }
                if(G.fin()==1)      //si partie arrétée
                    break;      //on sort
         }
}

void Partie::tour_du_joueur(int joueur)
{
            if(J[joueur-1].test_ia()==false)
                    jouer_humain(joueur);       //le joueur choisit son coup

            else if(J[joueur-1].test_ia()==true)
                    jouer_ia(joueur);
            G.initialiser_alignement();     //on reinitialise tous les alignements a 0
            for(int i=0; i<nb_joueur; i++)
            {
                    G.verification(J[i].renvoie_jeton(),i+1);       //on verifie les
                    alignements de chaque joueur
```

```
                    if(G.fin()==2)      //si un joueur a tout ses alignements
                    {
                            victoire(i+1);      //il gagne
                            break;      //si la partie d'arrete, on sort
                    }
                    else if(G.fin()==1)     //si la partie est arrétée
                    {
                            victoire(0);    //partie nulle
                            break;      //si la partie d'arrete, on sort
                    }
            }
            if(G.fin()==1)      //la partie est arretée
                    return;
            else if(G.nb_coup_possible()==0)        //la partie n'est pas arretée mais il
            n'y a plus de cases vides
                    victoire(0);        //donc partie nulle
}

void Partie::jouer_humain(int joueur_en_cours)
{
        if(limitateur_de_changement==joueur_en_cours)       //1 tour est passé depuis sa
        derniere utilisation
                limitateur_de_changement=0;     //on peut a nouveau echanger les pions
        string coup;    //coup choisit par le joueur
        jeton le_jeton;     //jeton du joueur
        le_jeton=J[joueur_en_cours-1].renvoie_jeton();      //on recupere le jeton du joueur
        qui joue
        G.affichage();
        if(joueur_en_cours==1)
                cout<< "\E[33;1mJOUEUR 1\E[m:";
        else if(joueur_en_cours==2)
                cout<< "\E[31;1mJOUEUR 2\E[m:";
        else if(joueur_en_cours==3)
                cout<< "\E[36;1mJOUEUR 3\E[m:";
        else if(joueur_en_cours==4)
                cout<< "\E[32;1mJOUEUR 4\E[m:";
        cout<<endl<<"choisissez votre coup:"<<endl;
        cout<<"     -taper un nombre pour placer un jeton dans la colonne
        correspondante"<<endl;
        cout<<"     -taper h ou a pour pivoter en sens horaire ou antihoraire"<<endl;
        if(limitateur_de_changement==0)     //pour eviter plusieur echanges a la suite (1
        seul pendant un tour complet)
        {
                if(nb_joueur==4)
                        cout<<"     -taper j1, j2, j3 ou j4 pour echanger les pions avec ce
                        meme joueur"<<endl;
                else if(nb_joueur==3)
                        cout<<"     -taper j1, j2 ou j3 pour echanger les pions avec ce meme
                        joueur"<<endl;
                else if(nb_joueur==2)
                        cout<<"     -taper j1 ou j2 pour echanger les pions avec ce meme
                        joueur"<<endl;
        }
        cout<<"     -taper s ou save pour sauvegarder la grille"<<endl;
        cout<<"     -taper stop pour arreter le jeu (partie nulle)"<<endl;
        cin>>coup;      //le joueur choisit son coup
        if(string_is_int(coup)==true && string_to_int(coup)!=0)     //si c'est un nombre
        different de 0
        {
                int nb_colonne=G.renvoie_colonne();     //on recupere le nombre de colonne
                int colonne=string_to_int(coup);    //on convertit le coup en int
                if(colonne>nb_colonne || G.colonne_remplie(colonne)==true)      //on
                compare, si la colonne demandée n'existe pas ou qu'elle est remplie
                        jouer_humain(joueur_en_cours);  //on recommence
                else
                        G.placer(colonne, le_jeton, joueur_en_cours);       //on place le
                        jeton
        }
        else if(string_is_int(coup)==false)     //si ce n'est un nombre
        {
                if(coup=="h" || coup=="horaire"||coup=="a"||coup=="antihoraire")
```

```
                //pour pivot, plusieurs orthographe autorisée
                {
                        string pivot=coup;
                        G.pivoter(pivot, le_jeton, joueur_en_cours);//on pivote
                }
                else
                if((((coup=="j1"||coup=="j2")&&nb_joueur>=2)||coup=="j3"&&nb_joueur>=3||coup==
                "j4"&&nb_joueur==4)&&limitateur_de_changement==0)        //pour un echange
                {
                        limitateur_de_changement=joueur_en_cours;       //le limitateur vaut
                        l'index du joueur qui joue
                        char forme_ori;     //forme du jeton de celui qui demande l'echange
                        char forme_dest;        //forme du jeton de celui qui subit l'echange
                        int couleur_ori;        //couleur du jeton de celui qui demande
                        l'echange
                        int couleur_dest;       //couleur du jeton de celui qui subit
                        l'echange
                        string echange=coup;
                        forme_ori=J[joueur_en_cours-1].renvoie_forme_jeton();       //on
                        recupere la forme du jeton
                        couleur_ori=J[joueur_en_cours-1].renvoie_couleur_jeton();       //on
                        recupere la couleur du jeton

                        if((coup=="j1"&&joueur_en_cours==1)||(coup=="j2"&&joueur_en_cours==2)|
                        |(coup=="j3"&&joueur_en_cours==3)||(coup=="j4"&&joueur_en_cours==4))
                                jouer_humain(joueur_en_cours);      //le joueur veut
                                échange avec lui meme
                        else
                        {
                                coup=coup.substr(1,2);      //on ne recuper de coup
                                seulement ce qui est entre le caractere 1 et 2 comprit,
                                c'est a dire ici le numero
                                int j=string_to_int(coup);      //on convertit en int
                                forme_dest=J[j-1].renvoie_forme_jeton();        //on
                                recupere la forme du jeton
                                couleur_dest=J[j-1].renvoie_couleur_jeton();        //on
                                recupere la couleur du jeton
                                G.echanger(forme_ori,forme_dest,couleur_ori,
                                couleur_dest);     //on fait enfin l'echange
                        }
                }
                else if(coup=="save"||coup=="s"||coup=="sauver"||coup=="sauvegarde")//pour
                sauvegarder
                {
                        G.sauvergarde(joueur_en_cours);     //on pratique a sauvegarde
                        jouer_humain(joueur_en_cours);      //une fois sauvegarder le joueur
                        peut de nouveau jouer
                }
                else if(coup=="stop")       //pour stopper la partie
                        G.impose_fin();     //si un joueur arrete la partie avant qu'elle
                        arrive a son therme c'est une partie nulle
                else      //entée invalide
                        jouer_humain(joueur_en_cours);      //on recommence
        }
}

void Partie::jouer_ia(int joueur_en_cours)
{
    int num_ia=joueur_en_cours-nb_joueur+(nb_IA-1);
    jeton le_jeton;     //jeton du joueur
    le_jeton=J[joueur_en_cours-1].renvoie_jeton();      //on recupere le jeton du joueur qui
    joue
    IA[num_ia].recupere_grille(G);
    int coup=IA[num_ia].min_max(joueur_en_cours);
    if(coup<=G.renvoie_colonne())
            G.placer(coup,le_jeton,joueur_en_cours);
    else if(coup==G.renvoie_colonne()+1)
            G.pivoter("horaire", le_jeton, joueur_en_cours);
    else if(coup==G.renvoie_colonne()+2)
            G.pivoter("antihoraire", le_jeton, joueur_en_cours);
}
```

```
////////////////// AUTRES METHODES DE LA CLASSE ////////////////////////

void Partie::victoire(int joueur)
{
        G.initialiser_alignement();     //on reinitialise les alignement
        for(int i=0; i<nb_joueur; i++)
                G.verification(J[i].renvoie_jeton(),i+1);       //on repratique toute les
                verifications
        system("clear");
        G.affichage();
        string choix;       //pour savoir si on recommence ou on arrete
        if(joueur==0)
        {
                cout<<endl<<"       *****************"<<endl;
                cout<<"     *   MATCH NUL   *"<<endl;
                cout<<"     *****************"<<endl;
        }
        else
        {
                cout<<endl<<"       *************************"<<endl;
                cout<<"     *   VICTOIRE DU JOUEUR "<<joueur<<" *"<<endl;
                cout<<"     *************************"<<endl;
        }
        cout<<"recommencer: taper 1"<<endl;
        cout<<"finir(quitter le jeu): taper 2"<<endl;
        cin>>choix;
        if(choix=="1")      //on recommence
        {
                G.preremplir_grille();      //on reconstruit la grille vide
                if(nb_IA>0)
                {
                        for(int i=0; i<nb_joueur; i++)
                                        G.recupere_jeton_joueur(i+1, jeton_joueur(i+1));
                        for(int i=nb_joueur-1; i>=nb_joueur-nb_IA; i--)
                        {
                                int num_ia=i-nb_joueur+nb_IA;
                                IA[num_ia].recupere_grille(G);
                                IA[num_ia].recupere_jeton();
                        }
                }
                tour_par_tour(1);       //on recommence la partie avec le joueur 1
        }
        else if(choix=="2")     //on arrete
                G.impose_fin();     //on simule une partie nulle
        else
        {
                cout<<"taper une commande valide s'il vous plait!"<<endl<<endl;
                victoire(joueur);
        }
}

jeton Partie::jeton_joueur(int joueur)
{
        return J[joueur-1].renvoie_jeton();
}

void Partie::charger_jeton_joueur()
{
        for(int i=0; i<nb_joueur;i++)
                J[i].charger_jeton(G.renvoie_jeton_joueur(i+1));
}
```