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

using namespace std;

////////////////////////// CONSTRUCTEUR ////////////////////////////////

ia::ia()
{
    difficulte=0;
    num_joueur=0;
    nb_joueur=0;
    nb_ia=0;
    nb_j=NULL;
    al=NULL;
    jetons_joueurs=NULL;
}

ia::~ia()
{
    delete[] jetons_joueurs;
    delete[] al;
    delete[] nb_j;
}

////////////////////// ALGORITHME PRINCIPAL ////////////////////////////

int ia::min_max(int joueur)
{

        grille G_pere(G_bis);       //on construit une grille pere
        num_joueur=joueur;      //num joueur vaut le joueur qu'on essaie de faire gagner
        int niveau=difficulte;      //le niveau de difficulté correspond a la profondeur
        d'analyse
        int colonne=0;
        int note=0;
        int coup=G_bis.renvoie_colonne();
        int coup_choisie=0;
        int note_max=-120000;  //on met a -infini
        int var=0;
        for(int i=0; i<coup+2; i++)
        {
                if(i<G_bis.renvoie_colonne())       //si on teste les placement
                        colonne=i+1;
                else      //si on teste les rotation
                        var++;
                if(G_bis.colonne_remplie(colonne)!=true || var==1||var==2)
                {
                        if(i<G_bis.renvoie_colonne())
                                G_bis.placer(colonne,jetons_joueurs[num_joueur-1],num_joueur);
                        else if(var==1)
                                G_bis.pivoter("horaire", jetons_joueurs[num_joueur-1],
                                num_joueur);
                        else if(var==2)
                                G_bis.pivoter("antihoraire", jetons_joueurs[num_joueur-1],
                                num_joueur);

                        G_bis.initialiser_alignement();     //on remet tous les alignements
                        a 0
                        for(int l=0; l<nb_joueur;l++)
                                G_bis.verification(jetons_joueurs[l],l+1);      //on fait
                                toutesles verifications pour chaque joueurs
                        for(int j=0; j<nb_joueur;j++)
                                al[j]=G_bis.renvoie_nb_alignement(j+1);     //on recupere
                                les alignements effectués de chaque joueurs
```

```
                        for(int k=0; k<nb_joueur;k++)
                                nb_j[k]=G_bis.renvoie_nb_jeton_aligne(k+1);     //on
                                recupere les jetons alignés de chaque joueur

                        if(joueur==nb_joueur)
                                note=min(niveau-1, 1, -120000, 120000);     //on appelle min
                                avec alpha=-infini et beta=+infini
                        else
                                note=min(niveau-1, joueur+1, -120000, 120000);
                        if(var==0)
                                G_bis.annule_placement(colonne);        //on recupere
                                l'ancienne grille donc on retire le coup effectué
                        else
                                G_bis.recopie(G_pere);      //on recupere l'ancienne grille
                                donc on retire le coup effectué
                        if(note>note_max||(note==note_max&&rand()%2==0))
                        {
                                note_max=note;
                                coup_choisie=i+1;
                        }
                }
        }
        return coup_choisie;
}

int ia::min(int niveau, int joueur, int alpha, int beta)
{
        G_bis.initialiser_alignement();
        for(int l=0; l<nb_joueur;l++)
                G_bis.verification(jetons_joueurs[l],l+1);
        for(int j=0; j<nb_joueur;j++)
                al[j]=G_bis.renvoie_nb_alignement(j+1);
        for(int k=0; k<nb_joueur;k++)
                nb_j[k]=G_bis.renvoie_nb_jeton_aligne(k+1);
        bool condition=0;
        for(int k=0; k<nb_joueur; k++)
        {
            if(G_bis.renvoi_nb_al_necessaire()==al[k])
                    condition=1;
        }
        if(niveau==0 ||(niveau!=0 && G_bis.grille_remplie()==true)||condition==1)
        {
                int nb=G_bis.renvoie_nb_jeton();
                int x=evaluation(joueur);

                if(x>0)
                    return x-nb;
                else if(x<0)
                    return x+nb;
                else
                    return x;
        }
        grille G_pere(G_bis);


        int colonne=0;
        int coup=G_bis.renvoie_colonne();
        int mini=120000;
        int note=0;
        int var=0;
        for(int i=0; i<coup+2; i++)
        {
                if(i<G_bis.renvoie_colonne())
                        colonne=i+1;
                else
                    var++;
                if(G_bis.colonne_remplie(colonne)!=true || var==1||var==2)
                {
                        if(i<G_bis.renvoie_colonne())
                                G_bis.placer(colonne,jetons_joueurs[joueur-1],joueur);
                        else if(var==1)
```

```
                                G_bis.pivoter("horaire", jetons_joueurs[joueur-1], joueur);
                        else if(var==2)
                                G_bis.pivoter("antihoraire", jetons_joueurs[joueur-1],
                                joueur);
                        G_bis.initialiser_alignement();
                        if(joueur+1==num_joueur)
                                note=max(niveau-1, joueur+1, alpha, beta);
                        else if(joueur==nb_joueur)
                                note=min(niveau-1, 1, alpha, beta);
                        else
                                note=min(niveau-1, joueur+1, alpha, beta);
                        if(note<mini||(note==mini&&rand()%2==0))
                                mini=note;
                        if(var==0)
                                G_bis.annule_placement(colonne);        //on recupere
                                l'ancienne grille donc on retire le coup effectué
                        else
                                G_bis.recopie(G_pere);      //on recupere l'ancienne grille
                                donc on retire le coup effectué
                        if(mini<=alpha)
                                return mini;
                        if(mini<beta)
                                beta=mini;
                }
        }
        return mini;
}

int ia::max(int niveau, int joueur, int alpha, int beta)
{
        G_bis.initialiser_alignement();
        for(int l=0; l<nb_joueur;l++)
                G_bis.verification(jetons_joueurs[l],l+1);
        for(int j=0; j<nb_joueur;j++)
                al[j]=G_bis.renvoie_nb_alignement(j+1);
        for(int k=0; k<nb_joueur;k++)
                nb_j[k]=G_bis.renvoie_nb_jeton_aligne(k+1);
        bool condition=0;
        for(int k=0; k<nb_joueur; k++)
        {
            if(G_bis.renvoi_nb_al_necessaire()==al[k])
                    condition=1;
        }
        if(niveau==0 ||(niveau!=0 && G_bis.grille_remplie()==true)||condition==1)
        {
                int nb=G_bis.renvoie_nb_jeton();
                int x=evaluation(joueur);

                if(x>0)
                    return x-nb;
                else if(x<0)
                    return x+nb;
                else
                    return x;
        }
        grille G_pere(G_bis);
        int colonne=0;
        int coup=G_bis.renvoie_colonne();
        int maxi=-120000;
        int note=0;
        int var=0;
        for(int i=0; i<coup+2; i++)
        {
                if(i<G_bis.renvoie_colonne())
                        colonne=i+1;
                else
                        var++;
                if(G_bis.colonne_remplie(colonne)!=true || var==1||var==2)
                {
                        if(i<G_bis.renvoie_colonne())
                                G_bis.placer(colonne,jetons_joueurs[num_joueur-1],num_joueur);
```

```
                        else if(var==1)
                                G_bis.pivoter("horaire", jetons_joueurs[num_joueur-1],
                                num_joueur);
                        else if(var==2)
                                G_bis.pivoter("antihoraire", jetons_joueurs[num_joueur-1],
                                num_joueur);
                        if(num_joueur==nb_joueur)
                                note=min(niveau-1, 1, alpha, beta);
                        else
                                note=min(niveau-1, joueur+1, alpha, beta);
                        if(note>maxi||(note==maxi&&rand()%2==0))
                                maxi=note;
                        if(var==0)
                                G_bis.annule_placement(colonne);        //on recupere
                                l'ancienne grille donc on retire le coup effectué
                        else
                                G_bis.recopie(G_pere);      //on recupere l'ancienne grille
                                donc on retire le coup effectué
                        if(maxi>=beta)
                                return maxi;
                        if(alpha>maxi)
                                alpha=maxi;
                }
        }
        return maxi;
}


int ia::evaluation(int joueur)
{
    G_bis.initialiser_alignement();
    for(int i=0; i<nb_joueur;i++)
    {
            if(G_bis.renvoi_nb_al_necessaire()<=al[i]&&(i+1)!=num_joueur)
                    return -120000;
            else if(G_bis.renvoi_nb_al_necessaire()<=al[i]&&(i+1)==num_joueur)
                    return 120000;
    }
    if(G_bis.fin()==1)
            return 0;
    int soustraction=0;
    for(int k=0; k<nb_joueur;k++)
    {
        if(k+1!=num_joueur)
                soustraction=soustraction-al[k];
    }
    int soustraction2=0;
    for(int k=0; k<nb_joueur;k++)
    {
        if(k+1!=num_joueur)
                soustraction2=soustraction2-nb_j[k];
    }
            return
            ((al[num_joueur-1]+soustraction)*10000)+(nb_j[num_joueur-1]+soustraction2)*1000;
}




////////////////////// AUTRES METHODES DE CLASSE ////////////////////

void ia::recupere_donnee_partie(int dif, int nb_joue, int ia)
{
    difficulte=dif;
    nb_joueur=nb_joue;
    nb_ia=ia;
    al=new int [nb_joueur];
    for(int i=0; i<nb_joueur;i++)
            al[i]=0;
    nb_j=new int [nb_joueur];
    for(int i=0; i<nb_joueur;i++)
```

```
            nb_j[i]=0;
}

void ia::recupere_jeton()
{
    jetons_joueurs=new jeton[nb_joueur];
    for(int i=0; i<nb_joueur; i++)
        jetons_joueurs[i]=G_bis.renvoie_jeton_joueur(i+1);
}

void ia::recupere_grille(grille g)
{
        G_bis.recopie(g);
}
```