Main.c

```
/**
* @file main.c
* @author Durand Thomas
* @brief
* @version 0.1
* @date 2019-10-15
*
* @copyright Copyright (c) 2019
*
*/

#include <stdio.h>
#include <stdlib.h>
#include "plateau.h"
#include "logique.h"
#include "rotation.h"
#include "constant.h"
/**
* @brief fonction main pour initialisation et deroulement du jeux
*
* @return int
*/
int main(){
int plateau[N][N];
int joueur,rep,testJouer,testRot;

printf("combien de rotaion voulez vous par joueuer?\n");
int rotation;
scanf("%d",&rotation);
int tabRotaion[2] = {rotation,rotation};
init(plateau);
joueur = 1;
while(1){
affichage(plateau);
do{
printf("joueur %d a toi de jouer, tu veux placer ton pion dans quel collone?",joueur);
scanf("%d",&rep);
testJouer = jouer(plateau,joueur,rep);
if (testJouer == 1){
printf("desole, il n'y a plus de place sur cette ligne\n");
}else if(testJouer == 2){
printf("desole, mais vous avez essayez de jouer hors du plateau\n");
```

```
}
}while(testJouer != 0);
if (aGagner(plateau) == 1){
affichage(plateau);
printf("Bravo joueur %d tu as gagner!!! \n",joueur);
exit(EXIT_SUCCESS);
}
affichage(plateau);
testRot = deroulementRotaion(plateau,joueur,tabRotaion[joueur-1]);
if (testRot == 1 && aGagner(plateau) == 1){
affichage(plateau);
printf("Bravo joueur %d tu as gagner!!! \n",joueur);
exit(EXIT_SUCCESS);
}

joueur = (joueur % 2) + 1;
}
return 0;
}
```

plateau.c

```
/**
* @file plateau.c
* @author Durand Thomas
* @brief ensemble des fonctions lie au plateau
* @version 0.1
* @date 2019-11-12
*
* @copyright Copyright (c) 2019
*
*/
#include <stdio.h>
#include <stdlib.h>

#include "constant.h"

/**
* @brief initialise le plateau de N*N a -1
*
* @param plateau
*/
void init(int plateau[N][N]){
for (int i = 0; i < N; i++){
for (int j = 0; j < N; j++){
plateau[i][j] = -1;
}
}
}
/**
* @brief fonction d'affichage du plateau de N*N
*
* @param plateau
*/
void affichage(int plateau[N][N]){

printf(" ");
for (int i = 0; i < N; i++){
printf("%d ",i);
}
printf("\n");
printf(" +");
for (int i = 0; i < N; i++){
```

```
printf("---+");
}
printf("\n");
for (int i = 0; i < N; i++){
printf("%d ",i);
for (int j = 0; j < N; j++){
if (plateau[j][i] == -1){
printf("| ");
}else{
printf("| %d ",plateau[j][i]);
}
}
printf("|\n");
printf(" +");
for (int i = 0; i < N; i++){
printf("---+");
}
printf("\n");
}
}

/**
* @brief joue un pion et le fait tomber tout en bas
*
* @param plateau
* @param joueur
* @param colone
* @return int
*/
int jouer(int plateau[N][N],int joueur, int colone){
if( colone < 0 || colone > N){
return 2;
}else if ((plateau[colone][0] != -1)){
return 1;
}

int i = 0;
while ((i < N) && (plateau[colone][i] == -1)){
i ++;
}
plateau[colone][i-1] = joueur;
return 0;
}
```

rotation.c

```
/**
* @file rotation.c
* @author Durand Thomas
* @brief ensemble des fonctions lie a la rotation
* @version 0.1
* @date 2019-11-12
*
* @copyright Copyright (c) 2019
*
*/
#include <stdio.h>
#include <stdlib.h>

#include "constant.h"

/**
* @brief applique une rotation de 90 degre a la matrice
*
* @param tableau
*/
void rotation(int tableau[N][N]){

for (int x = 0; x < N / 2; x++){
for (int y = x ; y < N - x - 1; y++){
int temp = tableau[x][y];
tableau[x][y] = tableau[y][N-1-x];
tableau[y][N-1-x] = tableau[N-1-x][N-1-y];
tableau[N-1-x][N-1-y] = tableau[N-1-y][x];
tableau[N-1-y][x] = temp;
}
}
}
/**
* @brief fais tomber un pions jusqu'a qu'il soit tout en bas
*
* @param tableau
* @param x
* @param y
* @param pion
*/
```

```
void descendre(int tableau[N][N],int x,int y,int pion){
tableau[x][y] = -1;
while ((y < N) && (tableau[x][y] == -1)){
y ++;
}
tableau[x][y-1] = pion;
}

/**
* @brief applique la gravite a tout les pions du plateau
*
* @param tableau
*/
void gravite(int tableau[N][N]){
for(int i = N-1;i >= 0; i--){
for(int j = N-1;j >= 0; j--){
descendre(tableau,i,j,tableau[i][j]);
}
}
}

/**
* @brief deroulement d'une rotation
*
* @param tableau
* @param joueur
* @param nbrRotation
* @return int
*/
int deroulementRotaion(int tableau[N][N],int joueur,int nbrRotation){
if (nbrRotation > 0 ){
int reponse;
printf("Voulez vous executer une rotaion? 1 pour oui 0 pour non");
scanf("%d",&reponse);
if(reponse == 1){
rotation(tableau);
gravite(tableau);
return 1;
}else{
return 0;
}
}
return 0;
}
```

logique.c

```
/**
* @file logique.c
* @author Durand Thomas
* @brief ensemble des fonctions de logique du jeux
* @version 0.1
* @date 2019-11-12
*
* @copyright Copyright (c) 2019
*
*/

#include <stdio.h>
#include <stdlib.h>

#include "constant.h"

/**
* @brief test si une ligne (verticale horizontale diagonale etc...) comporte 4 pions identique (condition de victoire)
*
* @param plateau
* @param x
* @param y
* @param sensx
* @param sensy
* @param score
* @param joueur
* @return int
*/
int testWin(int plateau[N][N],int x,int y, int sensx,int sensy,int score,int joueur){
x += sensx;
y += sensy;
if (score == 4){
return 1;
}else if((x < 0) || (y < 0) || (x > N) || (y > N) || (plateau[x][y] != joueur)){
return 0;
}else{
return testWin(plateau,x,y,sensx,sensy,score+1,joueur);
}
```

```
}

/**
* @brief applique la fonction testWin dans toutes les directions a tout les points
*
* @param plateau
* @return int
*/
int aGagner(int plateau[N][N]){
for (int i = 0; i < N; i++){
for (int j = 0; j < N; j++){
if(plateau[i][j] != -1){
for (int k = -1; k < 2; k++){
for (int l = -1; l < 2; l++){
if (((k != 0) || (l != 0)) && (testWin(plateau,i,j,k,l,1,plateau[i][j]) == 1)){
return 1;
}
}
}
}
}
}
return 0;
}
```