CAHIER DE CONCEPTION GÉNÉRALE

Projet d'algorithmique 2016-2017

Version: 1.8
Auteur: Romain JACQUIEZ, Antoine BERENGUER et Ouassim AKEBLI

ISEN Toulon - Yncrea
Maison du Numérique et de l'Innovation
Place Georges Pompidou
83000 Toulon

Description du document

Type	Version	Confidentialité
Cahier de conception générale	1.8	Usage externe

	Nom	Fonction	Date	Visa
Rédacteur	Romain JACQUIEZ Antoine BERENGUER Ouassim AKEBLI	Rédaction	04/01/17 au 17/01/17	
Vérificateur	Romain JACQUIEZ	Corrections	05/01/17 au 21/01/17	
Approbateur	Romain JACQUIEZ	Validation	21/01/17	

Destinataire	Fonction	Organisme
Client	Lecture	ISEN

Révisions du document

Version	Date	Rédacteur	Modifications
1.0	18/07/2016	FMC	Mise en forme
1.1	04/01/17	OA, RJ	Rédaction de contenu
1.2	04/01/17	RJ	Rédaction de contenu
1.3	05/01/17	AB	Suppression et ajout de prototypes
1.4	05/01/17	RJ	Ajout de l'architecture des fonctions
1.5	05/01/17	RJ	Ajout de fonctions
1.6	06/01/17	RJ	Ajout et correction des structures de données
1.7	10/01/17	RJ	Rajout des fonctions et arbre IA
1.8	17/01/17	RJ	Ajout fonction gérant les tours des joueurs

Sommaire

Index des illustrations

Index des tables

REFERENCES

Référence	Description	Nom
[1]		
[2]		

DEFINITIONS

Sans objet

ABBREVIATIONS

ISEN : Institut Supérieur de l'Electronique et du Numérique

1. INTRODUCTION

Le programme sera constitué de trois modules fonctionnels : le module principal, le module de calcul (le moteur) et le module d'affichage. Chaque module sera constitué d'un fichier source .c et si nécessaire d'un fichier d'en-têtes .h

2. MODULES FONCTIONNELS

A) Architecture des modules

Le module principal (main.c) fait le lien entre les deux autres modules (moteur.c et affichage.c) et gère le jeu en lui même.

Ce module récupère en outre les entrées faites par les joueurs.

Le module d'affichage gère toutes les opérations d'affichage à l'écran.

Le moteur gère tous les calculs liés aux coups des joueurs durant la partie.

B) Données utilisées par chaque module

Module principal :
- Toutes les données reçues au clavier ou à la souris
- Le tableau modélisant le plateau de jeu
- Les informations de chaque joueur
- Le statut de la partie (menu principal, partie en cours, partie terminée)

Module d'affichage :
- Le tableau modélisant le plateau de jeu
- Les informations de chaque joueur
- Le statut de la partie (menu principal, partie en cours, partie terminée)

Moteur :
- Le tableau modélisant le plateau de jeu

C) Échange de données entre modules

Le module principal communique avec le module d'affichage au niveau de l'état de la partie, de la progression du tour, des scores et de l'état du plateau.

Le moteur communique avec le module principal au niveau de l'état de la partie.

Le moteur et le module d'affichage ne communiquent pas.

3. STRUCTURES DE DONNÉES

A) Définition des structures de données

Informations des joueurs : structure de données
Contenu de la structure :
- Nom du joueur (facultatif)
- Symbole du joueur pendant la partie
- Score du joueur
- pointeur qui pointe vers le joueur suivant

Plateau de jeu : tableau d'entiers, deux dimensions, 5x5 cases
On aura aussi besoin d'un pointeur vers ce tableau

Les coordonnées des cases en mode graphique sont stockées dans un tableau 2D contenant 25 structures. Dans chaque structures, il y a 8 coordonnées correspondants au quatre coins de la case sur la grille.
On aura aussi besoin d'un pointeur vers ce tableau

B) Action portant sur ces structures de données

Structure des joueurs :

- Modification du nom du joueur quand renseigné
- Symbole utilisé par le joueur reste constant
- Score modifié quand quand réinitialisation ou victoire
- Joueur suivant reste constant

Plateau de jeu :

- Généré par le module principal
- Modifié quand signalé par le module principal

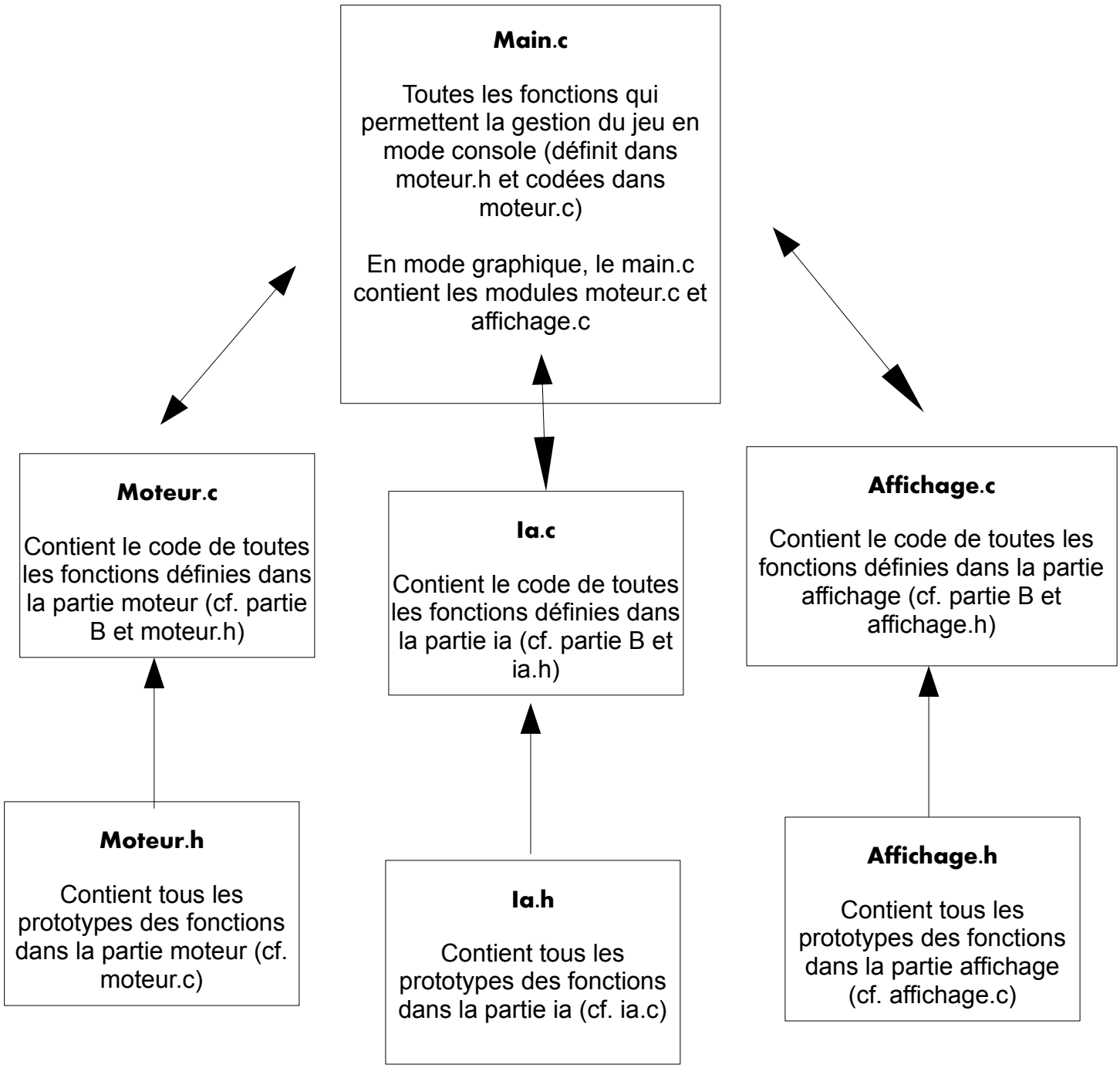
Le tableau des coordonnées des cases :
-Initialisé par une fonction avant le tracé de la grille en mode affichage
-Si modification de la fenêtre d'affichage, on doit refaire une initialisation car le programme se veux « responsive ».

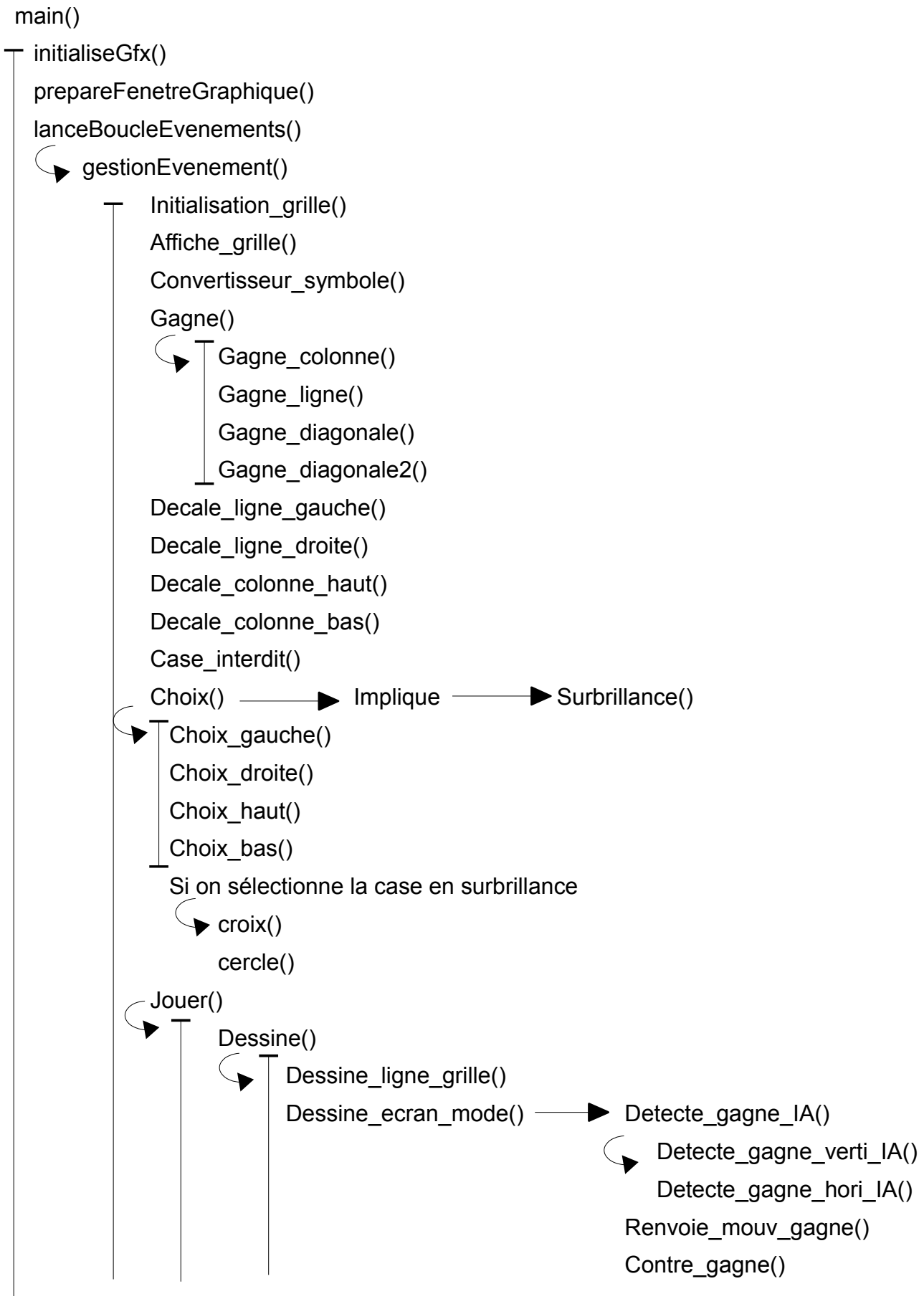
C) Visibilité des structures de données

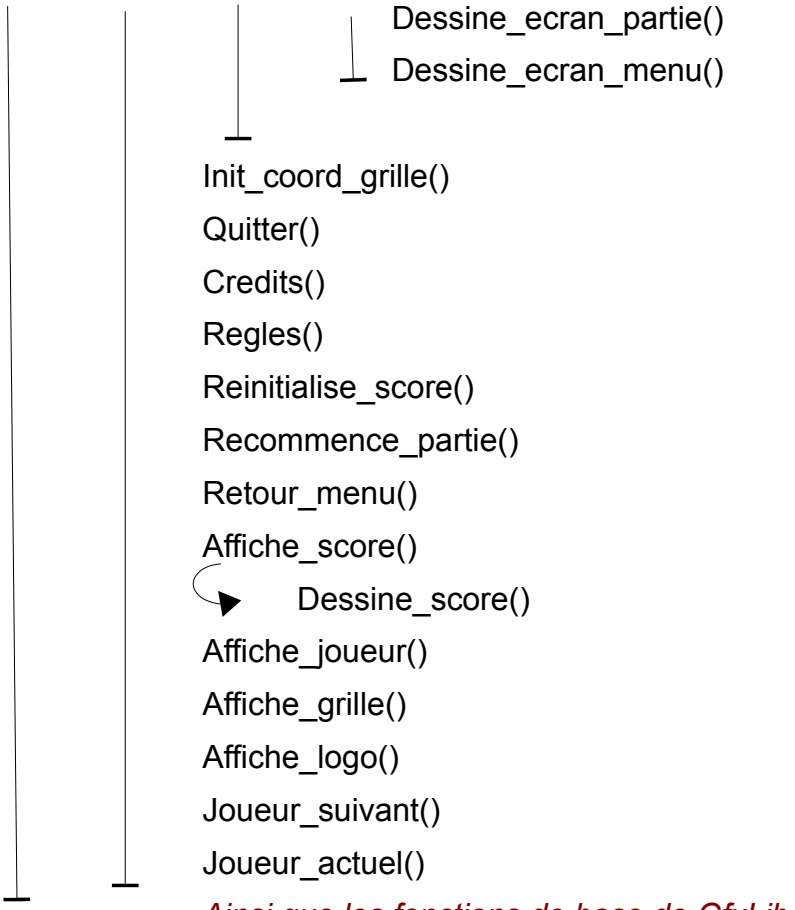
La structure d'infos des joueurs et le tableau sont locaux, dans la fonction de gestion d'événements.

4. ARBRE DES FONCTIONS ET FLUX DES DONNÉES

A) Arbre d'appel et flux de données




main()
initialiseGfx()
prepareFenetreGraphique()
lanceBoucleEvenements()
gestionEvenement()
Initialisation_grille()
Affiche_grille()
Convertisseur_symbole()
Gagne()
Gagne_colonne()
Gagne_ligne()
Gagne_diagonale()
Gagne_diagonale2()
Decale_ligne_gauche()
Decale_ligne_droite()
Decale_colonne_haut()
Decale_colonne_bas()
Case_interdit()
Choix()
Implique
Surbrillance()
Choix_gauche()
Choix_droite()
Choix_haut()
Choix_bas()
Si on sélectionne la case en surbrillance
croix()
cercle()
Jouer()
Dessine()
Dessine_ligne_grille()
Dessine_ecran_mode()
Detecte_gagne_IA()
Detecte_gagne_verti_IA()
Detecte_gagne_hori_IA()
Renvoie_mouv_gagne()
Contre_gagne()



Ainsi que les fonctions de base de GfxLib

La majorité de ces fonctions impliquent l'utilisation de Set() et Get() qui utilisent la fonction Cas_erreur().

B) Description des fonctions

Prototypes des fonctions du module moteur.c :

void Get(Grille, int, int)
Cette fonction permet d'accéder à la valeur d'une case d'un tableau dont les coordonnées sont passées en paramètre, en considérant les erreurs (dépassement de tableau etc).

Void Set(Grille,int,int,int)
Cette fonction permet de mettre valeur dans case d'un tableau dont les coordonnées sont passées en paramètre, en considérant les erreurs (dépassement de tableau etc).

bool Cas_erreur(int)
Cette fonction gère le cas d'erreur où la coordonnée n'est pas dans la grille.

void Initialisation_grille(Grille)
Cette fonction prend en paramètre un tableau d'entier et initialise toutes les cases à 0.

int Gagne(Grille, int joueur)
Cette fonction renvoie le numéro du joueur gagnant grâce aux fonctions ci-dessous.

int Gagne_colonne(Grille)
Cette fonction prend en paramètre un tableau et vérifie si un joueur a gagné grâce à un alignement sur une colonne.

int Gagne_ligne(Grille)
Cette fonction prend en paramètre un tableau et vérifie si un joueur a gagné grâce à un alignement sur une ligne.

int Gagne_diagonale(Grille)
Cette fonction prend en paramètre un tableau et vérifie si un joueur a gagné grâce à un alignement sur la diagonale.

int Gagne_diagonale2(Grille)
Cette fonction prend en paramètre un tableau et vérifie si un joueur a gagné grâce à un alignement sur la diagonale inverse.

void Joueur_suivant()
Cette fonction passe au joueur suivant (change la direction du pointeur général vers le joueur qui doit jouer)

int Joueur_actuel()
Cette fontion renvoie le numéro du joueur dont c'est le tour de jouer

void Decale_ligne_gauche(*Grille, int ligne, int indice)
Cette fonction prend en paramètre l'indice d'une ligne d'un tableau et permet de décaler la valeur de toutes les cases sur la longueur de cette ligne d'un cran vers la gauche.

void Decale_ligne_droite(*Grille, int ligne, int indice)
Cette fonction prend en paramètre l'indice d'une ligne d'un tableau et permet de décaler la valeur de toutes les cases sur la longueur de cette ligne d'un cran vers la droite.

void Decale_colonne_haut(*Grille, int ligne, int indice)
Cette fonction prend en paramètre l'indice d'une colonne d'un tableau et permet de décaler la valeur de toutes les cases sur la longueur de cette colonne d'un cran vers le haut.

void Decale_colonne_bas(*Grille, int ligne,int indice)
Cette fonction prend en paramètre l'indice d'une colonne d'un tableau et permet de décaler la valeur de toutes les cases sur la longueur de cette colonne d'un cran vers le bas.

int Case_interdit(Grille, struct)
Cette fonction prend en paramètre un tableau et des coordonnées et renvoie un booléen qui détermine si ces coordonnées correspondent à une case à l'extimité du tableau ou non.

struct *Choix(struct)*
Cette fonction détermine grâce aux autres fonctions ci-dessous, quelles sont les cases où l'on peut placer le cube et ensuite décaler.

*struct Choix_gauche(struct)
Cette fonction prend en paramètre les coordonnées d'une case et permet de déterminer quelle case à gauche de celle-ci on peut s'y déplacer.

*struct Choix_gauche(struct)
Cette fonction prend en paramètre les coordonnées d'une case et permet de déterminer quelle case à droite de celle-ci on peut s'y déplacer.

*struct Choix_haut (struct)
Cette fonction prend en paramètre les coordonnées d'une case et permet de déterminer quelle case en haut de celle-ci on peut s'y déplacer.

*struct Choix_bas(struct)
Cette fonction prend en paramètre les coordonnées d'une case et permet de déterminer quelle case en bas de celle-ci on peut s'y déplacer.

Prototypes des fonction du module affichage.c :

void Affiche_grille(Grille)
Cette fonction prend en paramètre un tableau d'entier et utilise deux variables local i et j

La fonction permet d'afficher la grille du Quixo en mode console.

void Quitter(struct)
Cette fonction prend des coordonnées en paramètre
Elle permet de quitter le programme si on clique sur le bouton quitter

void Credits(struct)
Même principe mais pour les credits

void Jouer(struct)
Même principe mais pour jouer

void Regles(struct)
Même principe mais pour les régles

void Reinit_score(struct)
Même principe mais pour réinitialiser les scores

void Recommence_partie(struct)
Même principe mais pour recommencer une partie

void Retour_menu(struct)
Même principe mais pour revenir au menu

void Affiche_score(int,int)
Fonction qui prend en paramètre deux entiers (les scores de chaque joueur) pour les afficher en mode graphique

void Affiche_joueur(int)
Fonction qui prend en paramètre un entier (le numéro du joueur) pour l'afficher quel joueur doit jouer ce tour

void Affiche_logo()
Fonction qui affiche le logo du jeu en mode graphique

void croix(pointeur vers un tableau 2D de struct ,struct)
Fonction qui trace une croix sur la case passé en paramètre

void cercle(pointeur vers un tableau 2D de struct ,struct)
Fonction qui trace une cercle sur la case passé en paramètre

void surbrillance(pointeur vers un tableau 2D de struct ,struct)
Fonction qui trace fait une surbrillance sur la case passé en paramètre

void Init_coord_grille(pointeur vers un tableau 2D de struct)
Fonction qui initialise le tableau de structure avec toutes les coordonnées

**struct Detecte_case_clic(int,int,int,int,int,int)*

Fonction qui renvoie un pointeur vers une structure qui contient les coordonnées de la case où l'utilisateur à cliqué.

void Dessine(int)
Fonction qui appelle les fonctions de dessin nécessaires en fonction de l'avancement de la partie (passé en paramètre sous forme d'entier)

void Dessine_ligne_grille(tableau 2D de struct)
Fonction qui dessine les ligne de la grille en fonction des coordonnées des structs qui sont dans le tableau 2D de structs

void Dessine_ecran_partie ()
Fonction qui dessine en graphique la partie

void Dessine_ecran_menu()
Fonction qui dessine en graphique le menu

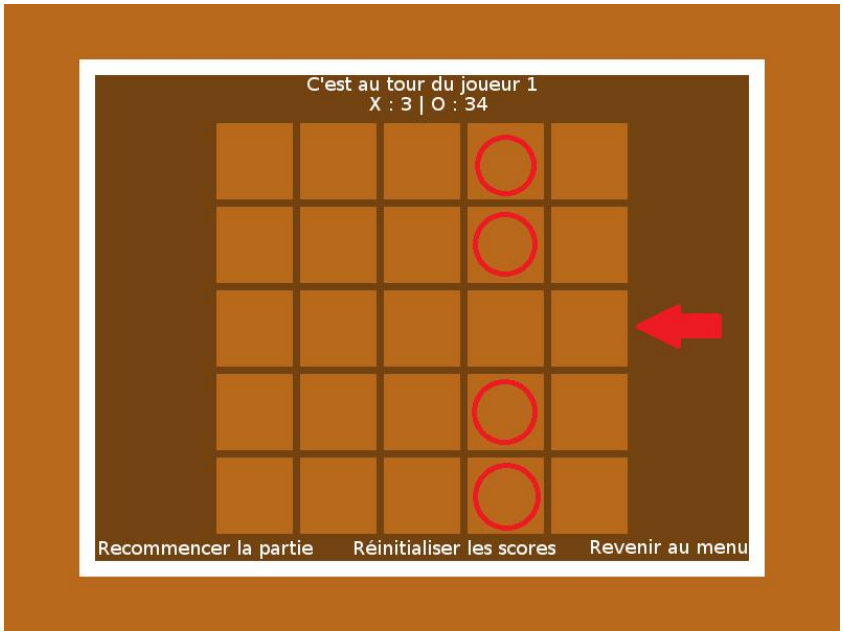
void Dessine_ecran_mode()
Fonction qui dessine en graphique l'écran des modes

char Convertisseur_symbole(int)
Fonction qui transforme un entier en un char pour l'affichage convivial du tableau en mode console

Prototypes des fonctions du module ia.c :

int Detecte_gagne_IA(Grille)
Fonction qui renvoie l'inde de la colonne ou de la ligne où ce joue le déplacement gagnant

int Detecte_gagne_verti_IA(Grille)
Fonction qui renvoie l'indice de la ligne ou un coup peut permettre de gagner :



int Detecte_gagne_hori_IA(Grille)
Fonction qui l'indice de la ligne ou un coup peut permettre de gagner

int Renvoie_mouv_gagne(Grille)
Fonction qui renvoie quel type de déplacement est a faire sur la ligne/colonne renvoyée par les fonctions précédentes, pour gagner la partie

int Contre_gagne(Grille)
Fonction qui renvoie quel type de déplacement est a faire sur la ligne/colonne renvoyée par les fonctions précédentes, pour empêcher le joueur de gagner la partie