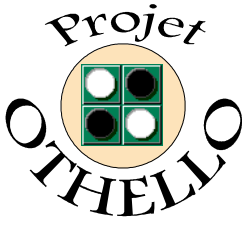


Conception & développement informatique B0

CNAM Versailles

Document de Conception Détaillée Indice A

SOMMAIRE

Indice	Date de diffusion	Intitulé
A	02/06/1997	Document initial

Toute modification de ce document par rapport à sa version précédente est repérée par un trait vertical dans la marge gauche.

1. OBJET :

Objet : Ce document constitue le Document de Conception Détaillée du logiciel OTHO dans le cadre du projet OTHELLO.

Domaine d'application : Logiciel destiné à une application type grand public.

Documents de référence : Cahier des Charges indice B du 22/03/1997.
Spécification Technique de Besoin indice A du 22/03/1997.
Document de Conception Préliminaire indice A du 26/04/1997.

2. INTRODUCTION :

Ce document permet de définir pour la phase de codage les modules réalisant les fonctionnalités du logiciel OTHO. Les fonctions de bas niveau, et donc très proche du langage de programmation, ne sont pas ici retranscrites (elles seraient le reflet parfait du code !!). En outre, ce DCD a pour but aussi de présenter les différents algorithmes de réflexion mis en oeuvre dans notre logiciel.

3. ALGORITHMES DE REFLEXION MIS EN OEUVRE :

Comme convenu dans les spécifications, notre logiciel doit disposer de plusieurs niveaux de difficulté pour un jeu Humain / Ordinateur. Ainsi, grâce aux informations que nous avons récupérés d'Internet sur les divers jeux existants, et aussi pour « corser » la difficulté, nous avons choisi d'utiliser plusieurs algorithmes en fonction des niveaux. Ce choix s'est imposé à nous surtout en terme de pédagogie pour les joueurs débutant (perdre à chaque fois lorsqu'on apprend un jeu n'est pas du plus motivant !!). De plus, cela nous permettait d'avoir plusieurs « vues » sur les tactiques du jeu OTHELLO.

3.1 Algorithme « Périmètre » : pour le niveau initiation

C'est un algorithme assez faible. Il est basé sur la priorisation de 4 zones importantes du damier comme indiqué dans le schéma ci-dessous (ordre de priorité croissante). La dernière zone grisée étant le « reste à jouer ».

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
1	1		3	3	3	3		1
2			4	4	4	4		
3	3	4	2	2	2	2	4	3
4	3	4	2			2	4	3
5	3	4	2			2	4	3
6	3	4	2	2	2	2	4	3
7			4	4	4	4		
8	1		3	3	3	3		1

C'est un algorithme cherchant à obtenir le meilleur placement pour les pions de l'ordinateur et non pas à maximiser le nombre de pions sur le damier. C'est là sa faiblesse.

3.2 Algorithme « Stratégique » : pour le niveau débutant

Par rapport au précédent algorithme, celui-ci utilise une fonction de priorisation plus intéressante, et recherche le meilleur score obtenu pour une même famille de cases. Le détail des priorités est défini dans le schéma de la page suivante :

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
1	1	8	2	4	4	2	8	1

2	8	9	7	6	6	7	9	8
3	2	7	3	5	5	3	7	2
4	4	6	5			5	6	4
5	4	6	5			5	6	4
6	2	7	3	5	5	3	7	2
7	8	9	7	6	6	7	9	8
8	1	8	2	4	4	2	8	1

C'est encore un algorithme de niveau faible car il ne sait pas anticiper les coups de l'adversaire et les pièges qui peuvent lui être tendus.

3.3 Algorithme α-β : pour les niveaux « Intermédiaire - Confirmé et Expert ».

C'est l'algorithme le plus puissant qui existe actuellement. Nos recherches s'étaient d'abord orientées vers l'algorithme MIN-MAX mais nos essais pendant le codage se sont révélés très utile pour la convivialité de notre jeu : en effet, nous arrivions à une réflexion très lente de la part de l'ordinateur : plus d'un milliard de feuilles (de solutions de jeu). Le principe d'$\alpha-\beta$ est un MIN-MAX amélioré restreignant la zone de recherche du coup vers le meilleurs coups (coupe de branches de recherche)

L'annexe n° 1 fourni le document « Lothello Version 1.1 » qui a été à la base de nos travaux sur l'algorithme de réflexion d'OTHO. Ce document est très détaillé et permet de mieux comprendre la philosophie de MIN-MAX et α-β car, hélas, comme pour beaucoup d'autre programmes d'OTHELLO, les sources ne sont pas fournis.

4. DESCRIPTION DES MODULES :

4.1 Principe :

Chaque module est décrit de la manière suivante :

- NOM DU MODULE
  Appellation du module dans le code source

- ROLE
  En quelque mot la fonction de base qu'il réalise

- ENTREES / SORTIES
  Interface de communication avec les autres modules du code source

  FONCTIONNEMENT
  Permet de décrire de façon soit succincte ou sous forme de pseudo-code, les traitements qui sont déroulés à l'intérieur du module.

La description des fonctions de bas niveau, comme l'affichage de textes sous forme de dialogue, la gestion des différents curseurs souris, la lecture des fichiers, etc n'est pas réalisé dans ce document. En effet, la plupart de ces fonctions impliquent directement le langage de programmation et les « astuces » ou les utilisation de librairies liées à ce langage. Inclure ce type de fonction dans un tel document serait une pure perte de temps, et de plus complètement inutile. En outre, et pour conclure, un Document de Conception Détaillée se doit encore d'être indépendant (ou assez indépendant) du langage de programmation.

4.2 Descriptions :

Vous trouverez dans les pages ci-après les descriptions des principaux modules composant le logiciel OTHO.

Module :	**otho (programme principal)**
Rôle :	programme principal (le « main » en C) : squelette de l'application
Entrées :	-
Sorties :	-

Fonctionnement :

initialisation de toutes les fonctions utilisées dans la librairie

test et initialisation du matériel : **materiel ()**

test et initialisation de la mémoire : **allocation_memoire ()**

initialisation des variables globales

affichage de l'écran d'accueil : **ecran_presentation ()**

appel du menu principal (sortie uniquement à la fin du jeu) : **menu_principal ()**

restauration du contexte graphique : fin d'utilisation de la souris, libération de la mémoire, ...
Affichage d'un écran de fin : réalisation - concepteurs - money, money, money ...

Module :	**materiel**
Rôle :	Fonction permettant de vérifier si l'utilisateur d'OTHO possède bien le matériel requis pour l'utilisation du logiciel.
Entrées :	-
Sorties :	-

Fonctionnement :

Test de la carte graphique :
Si erreur (carte non reconnue) : arrêt de l'application
Affichage à l'écran du type de carte

Test de la quantité de mémoire vidéo
Si erreur (mémoire insuffisante) : arrêt de l'application

Détermination du type de processeur
Si erreur (386 minimum) : arrêt de l'application

Détermination de la présence de la souris
Si erreur (pas de souris) : arrêt de l'application
Récupération des informations souris (utilisation click - captures, ...)
Affichage à l'écran des informations souris.

Module :	**allocation_memoire**
Rôle :	allouer de la mémoire pour les éléments graphiques de l'application
Entrées :	-
Sorties :	-

Fonctionnement :

Allocation de mémoire : pour les chiffres graphiques (score) + chiffre graphique tampon
pour les pions de jeu + pion tampon
pour les indicateurs de choix

Si erreur : arrêt de l'application

Chargement en mémoire des high-scores d'OTHO : **Charge_score ()**

Module :	**ecran_presentation**
Rôle :	Fonction permettant d'afficher l'écran de présentation (image de R. Gatliff)
Entrées :	-
Sorties :	-

Fonctionnement :

Chargement de l'image b1024.gif - b80x60b.gif b64x48b.gif dans le répertoire « images » en fonction de la résolution de l'écran

Erreur en cas de problème de mode vidéo et arrêt de l'application

Afficher le curseur de la souris

Attente de quelques secondes ou possibilité de passer soit en cliquant soit en appuyant sur une touche

Disparition de l'image

Module :	**menu_principal**
Rôle :	Fonction permettant d'afficher et de gérer le menu principal.
Entrées :	-
Sorties :	-

Fonctionnement :

Chargement de l'image menu1.gif (image de fond normale) et menu2.gif (image de fond avec « surbrillance ») dans le répertoire images.

Affichage et gestion de la souris:
Capture d'écran sur click droit de la souris
Prise en compte des coordonnées du curseur de la souris en X et Y
Prise en compte des zones de validité pour la prise en compte des options et l'affichage en surbrillance de l'option sur passage de la souris dans la zone.
Prise en compte du click gauche pour la sélection des options.

Gestion du clavier :
Prise en compte des appuis clavier (↑ ↓ ENTREE)
Prise en compte des zones de validité pour la prise en compte des options et l'affichage en surbrillance de l'option sur passage de la souris dans la zone.

Prise en compte des choix d'option du menu principal tant que pas choix Quitter
- Initiation : **jouer** (INITIATION)
- Jouer : **jouer** (NORMAL)
- Visualiser : **Visualiser ()**
- A propos : **apropos** ()
- Règles : **regles** ()
- Ouvertures : **ouverture** ()
- Scores : **affiche_high_score** (ECRAN_MENU_GENERAL)

Module :	**jouer**
Rôle :	Fonction permettant le jeu à plusieurs à Othello
Entrées :	-
Sorties :	-

Fonctionnement :

initialise_ecran_jeu ()

détermination du type de jeu à mettre en place
Si Normal : **jeu_normal ()**
Si Initiation **: initiation ()**

Module :	**initialise_ecran_jeu**
Rôle :	Fonction permettant d'initialiser l'écran de jeu (mise en place des affichages, ...)
Entrées :	-
Sorties :	-

Fonctionnement :

Chargement de l'image « \images\otho.gif » avec effet d'affichage progressif

Initialisation des objets graphiques du jeu :
Chiffres indiquants les scores
Pions noirs et blancs (définition de leur placement sur otho.gif et masquage)

Affichage de la zone d'aide en ligne en bas de l'écran

Module :	**jeu_normal**
Rôle :	Fonction permettant de jouer en mode normal à OTHO.
Entrées :	-
Sorties :	-

Fonctionnement :

Chargement de l'image « \images\otho.gif » avec effet d'affichage progressif

Affichage de la boîte de dialogue de choix du type de jeu
Si choix = Annuler : retour au menu principal.
initialise_jeu ()

Cas d'une partie sauvegardée alors
affichage d'une boite de dialogue saisie du Nom de la partie grâce au clavier
Affichage en aide en ligne du chargement de la partie en cours
Charge_partie (nom de la partie**)**

Initialisation des paramètres de jeu : type de jeu = **choix_Joueur ()**
si type de jeu = jeu à deux joueurs
jeu_2joueurs ()
Si type de jeu = ordinateur contre ordinateur
ordinateur_vs_ordinateur ()

Module :	**jeu_2joueur**
Rôle :	Fonction permettant de joueur à deux joueurs (humain / ordinateur)
Entrées :	-
Sorties :	-

Fonctionnement :

Test de reprise de partie négatif

Saisie des noms des joueurs humains
- Si Humain/humain : affichage d'une boîte de dialogue et saisie du nom du joueur avec les pions noirs et du joueur avec les pions blancs
- Si OTHO noir : affichage d'une boîte de dialogue et saisie du nom du joueur avec les pions blancs
- Si OTHO Blanc : affichage d'une boîte de dialogue et saisie du nom du joueur avec les pions noirs

Affichage du nom des joueurs dans leur fenêtres respectives

Si test de reprise de partie négatif alors
- début de jeu normal
- Affichage en aide en ligne que le Joueur Noir commence
- **statistique** (Joueur Noir)

Sinon
- Affichage du dernier coup de la partie reprise : **coup_suivant** ()
- Affichage en aide en ligne : reprise d'une partie

Test initiation (affichage des coups possibles)

Tant qu'on ne quitte pas le jeu

Initialise_tableau_commande ()

Si c'est à l'ordinateur de jouer :

Appels des fonctions pour que l'ordinateur joue un coup et calcule les statistiques selon les niveaux de jeu choisis :

Si niveau initiation
- **ordinateur_debutant** (couleur de l'ordinateur)

Si niveau Débutant
- **ordinateur_intermediaire** (couleur de l'ordinateur)

Si niveau intermédiaire
- **ordinateur_expert** (couleur de l'ordinateur, profondeur_intermédiaire)

Si niveau confirmé
- **ordinateur_expert** (couleur de l'ordinateur, profondeur_confirmé)

Si niveau expert=
- **ordinateur_expert** (couleur de l'ordinateur, profondeur_expert)

Calcul des statistiques : **statistique** (couleur de l'ordinateur)

Affichage pions : **affiche_pions** ()

Mémorise jeu courant : **memorise_jeu** (couleur du joueur courant)

Changement de main : **change_joueur** ()

Sinon c'est au joueur humain de jouer :

Si niveau initiation :

affichage des coups possibles : **affiche_coup** (couleur du joueur courant)

Attente click souris sur événements :
- bouton coup précédent : **coup_precédent ()**
- bouton coup suivant : **coup_suivant ()**

bouton conseil : **affiche_coup** (couleur du joueur courant)
conseil_coup (couleur du joueur courant)
bouton stop : définition de la fin du jeu : fin de jeu = OK
click sur damier :
détermination de l'endroit
Vérification si emplacement autorisé alors
on valide le choix et on affiche le coup joué
jouer_coup (X, Y, couleur du joueur courant)
Calcul des statistiques : **statistique** (couleur du joueur courant)
affichage : **affiche pions ()**
mémorisation du jeu courant :
memorise_jeu (couleur du joueur courant)
changement de main : **change_joueur** ()
sinon
Affichage dans l'aide en ligne : coup impossible

tests de fin de jeu
si les deux joueurs doivent passer leur tour
Affichage à l'écran : fin du jeu !!
fin du jeu = OK
ou
si le joueur courant doit passer son tour
Affichage fenêtre d'information : le joueur doit passer son tour
changement de main : **change_joueur ()**
ou
Tests partie terminée
Congratulations au gagnant : **fin_jeu ()**
Gestion des high-scores : **high_scores ()**
fin du jeu = OK
Jusqu'à fin du jeu = OK.

Sauvegarde de la partie jouée si au moins un coup a été joué et si le niveau de jeu contre l'ordinateur est différent de l'initiation.
si jeu humain contre humain : **sauve_partie ()**
sinon (jeu humain contre ordinateur sauf niveau initiation) : **sauve_partie ()**

Module :	**ordinateur_vs_ordinateur**
Rôle :	Fonction permettant à deux ordinateurs de jouer entre eux à des niveaux quelconques
Entrées :	-
Sorties :	-

Fonctionnement :

Test si reprise partie = non : nom des joueurs = OTHO BLANC et OTHO NOIR
Affichage du nom des joueurs dans leur fenêtres respectives
Affichage en aide en ligne :Le Joueur Noir commence
statistique (Joueur Noir)
Sinon
Affichage du dernier coup de la partie reprise : **coup_suivant** ()
Affichage en aide en ligne : reprise d'une partie

Tant qu'on ne quitte pas le jeu

Initialise_tableau_commande ()

Si click souris sur le bouton « suivant » :
Appels des fonctions pour que l'ordinateur joue un coup et calcule les statistiques selon les niveaux de jeu choisis :
Si niveau Débutant
ordinateur_intermediaire (couleur du joueur courant)
Si niveau intermédiaire
ordinateur_expert (couleur du joueur courant, profondeur_intermédiaire)
Si niveau confirmé
ordinateur_expert (couleur du joueur courant, profondeur_confirmé)
Si niveau expert=
ordinateur_expert (couleur du joueur courant, profondeur_expert)

Calcul des statistiques : **statistique** (couleur du joueur courant)

Affichage pions : **affiche_pions** ()

Mémorise jeu courant : **memorise_jeu** (couleur du joueur courant)

Changement de main : **change_joueur** ()

Attente click souris sur événements :
- bouton coup précédent : **coup_precédent ()**
bouton conseil : **affiche_coup** (couleur du joueur courant)
conseil_coup (couleur du joueur courant)
bouton stop : définition de la fin du jeu : fin de jeu = OK

tests de fin de jeu
si les deux joueurs doivent passer leur tour
Affichage à l'écran : fin du jeu !!
fin du jeu = OK
ou
si le joueur courant doit passer son tour
Affichage fenêtre d'information : le joueur doit passer son tour
changement de main : **change_joueur ()**
ou
Tests partie terminée
Congratulations au gagnant : **fin_jeu ()**
Gestion des high-scores : **high_scores ()**
fin du jeu = OK
Jusqu'à fin du jeu = OK.

Sauvegarde de la partie jouée si au moins un coup a été joué : **sauve_partie ()**

Module :	**initialise_jeu**
Rôle :	Fonction permettant d'initialiser le jeu
Entrées :	-
Sorties :	-

Fonctionnement :

Mise à zéro de toutes les variables globales concernant :
- Appels = 0
- Données de statistiques pour joueurs Noir et Blancs

Préparation des tableaux de bord :
Affichage de la boîte à outils
Affichage des tableaux de score joueurs BLANC et NOIR
Affichage de la zone d'aide en ligne
Affichage des pions dans les zones réservées des tableaux de score joueurs et dans la boite à outils

Initialisation du damier : damier = vide

Placement des pions en position début de jeu :
jouer_coup (4,4, BLANC**) jouer_coup (**4,5, NOIR**) jouer_coup (**5,5, BLANC**) jouer_coup (**5,4, NOIR**)**

Affichage du nombre de pions à l'écran : **compte_pions ()**

Affichage des pions sur le damier : **affiche_pions ()**

Mémorisation du jeu courant

Module :	**initialise_tableau_commande**
Rôle :	Fonction initialisation le tableau de commande du jeu (boîte à outils)
Entrées :	-
Sorties :	-

Fonctionnement :

Affichage (de type bouton relâché) des boutons « < », « > », « stop », « conseil »
(la gestion des boutons doit être réalisée de manière 3D : enfoncé / relâché)

Affichage de la version d'OTHO

Module :	**compte_pions**
Rôle :	Fonction permettant de compter les nombre de pions de chaque joueur
Entrées :	-
Sorties :	-

Fonctionnement :

NB_Noirs = 0
NB_Blancs = 0

Balayage du damier en X et Y
    Si pion blanc : NB_Blancs = NB_Blancs + 1
    Si pion noir : NB_Noirs = NB_Noirs + 1
Fin balayage

Affichage grâce aux nombre graphiques du nombre de pions noirs et blancs.

Mise à jour des statistiques :
Statistiques_NOIR_Score = NB_Noirs
Statistiques_BLANC_Score = NB_Blancs

Module :	**affiche_pions**
Rôle :	Fonction permettant d'afficher un pion sur le damier avec un effet de retournement
Entrées :	-
Sorties :	-

Fonctionnement :

Balayage en X et Y du damier pour connaître l'emplacement des nouveaux pions.
    Affichage du nouveau pion avec effet retournement (5 « images » à prendre sur le damier originel)
fin balayage

Module :	**choix_joueur**
Rôle :	Fonction permettant de laisser choisir qui des deux joueurs commence en premier.
Entrées :	-
Sorties :	TypeJeu

Fonctionnement :

Détermination du type de jeu :
    Si Ordinateur contre ordinateur : typeJeu = 1 sinon typejeu = 0
    Si HumainNoir contre ordinateur Blanc : CouleurOrdinateur = BLANC
    Si Humain Blanc contre ordinateur Noir : CouleurOrdinateur = NOIR

Module :	**change_joueur**
Rôle :	Fonction permettant de changer de main au cours du jeu.
Entrées :	-
Sorties :	-

Fonctionnement :

JoueurCourant = JoueurAdverse (passage à l'autre joueur)

Affichage du joueur courant dans la zone de la boîte à outils (retournement du pion) : **affiche_pions ()**

Module :	**jouer_coup**
Rôle :	Fonction permettant de jouer un coup.
Entrées :	caseX caseY : coordonnées du coup joué. Couleur : couleur des pions du joueur
Sorties :	-

Fonctionnement :

Mise à jour du damier avec le nouveau coup : damier (caseX, Case Y) = Couleur

Affichage du coup joué : affichage d'un pion aux coordonnées CaseX, CaseY

Module :	**coup_précedent**
Rôle :	Fonction permettant de gérer le clic souris sur le bouton coup précédent.
Entrées :	-
Sorties :	-

Fonctionnement :

Si on n'est pas au premier coup de la partie :
CouleurJoueur = couleurJoueurPrécédent (changement de couleur et donc de joueur)
Reprise du damier (-1)

ré-affectation des statistiques du damier (-1): score, mobilité, évaluation, feuilles, profondeur, coups

balayage du damier en X et Y
jouer_Coup (caseX, case Y, damier (-1) (CaseX, CaseY)**)**
Fin balayage

statistique (couleurJoueur**)**

change_Joueur ()

sinon
Afficher dans l'aide en ligne : « Vous ne pouvez pas revenir en arrière »

Module :	**coup_suivant**
Rôle :	Fonction permettant de gérer le clic souris sur le bouton coup suivant.
Entrées :	-
Sorties :	-

Fonctionnement :

Si on n'est pas au dernier coup de la partie en cours (et non pas dernier coup de la partie)
CouleurJoueur = couleurJoueurSuivant (changement de couleur et donc de joueur)
Reprise du damier actuel (+1)

ré-affectation des statistiques du damier (+1): score, mobilité, évaluation, feuilles, profondeur, coups

balayage du damier en X et Y
jouer_Coup (caseX, case Y, damier (-1) (CaseX, CaseY)**)**
Fin balayage

statistique (couleurJoueur**)**

change_Joueur ()

sinon
Affichage dans l'aide en ligne : Vous ne pouvez pas avancer !!

Module :	**memorise_jeu**
Rôle :	Fonction permettant de mémoriser les coups du jeu.
Entrées :	JoueurCourant : couleur du joueur demandant de recalculer les statistiques
Sorties :	-

Fonctionnement :

copie du damier en cours en damier (N°_du_coup) avec couleur = JoueurCourant
N° du coup = N°_du_Coup +1

idem pour les statistiques : (enregistrement mémoire des statistiques du damier en cours)

Module :	**conseil_coup**
Rôle :	Fonction permettant de gérer le clic souris sur le bouton « Conseil ».
Entrées :	Joueur_Courant
Sorties :	-

Fonctionnement :

NOTA : le conseil est basé sur une recherche du meilleur coup avec un niveau expert.:

Calcul du nombre de pions : Statistique_NOIR_Score + Statistiques_BLANC_Score
Détermination de la profondeur : profondeur = 5 (profondeur au niveau expert)

recherche du meilleur coup avec la méthode alpha-beta :
alpha-beta (Damier, profondeur, JoueurCourant, X, Y, ScoreCoup, note = 0, qui = « »)

Affichage du coup conseillé dans l'aide en ligne : « Je vous conseille la case »,X;Y
Appels = 0

Module :	**retourne**
Rôle :	Fonction permettant de déterminer si le placement est autorisé et/ou permet de retourner les pions.
Entrées :	Damier : damier de jeu sur lequel on travaille X, Y : coordonnées du pion placé flip : autorisation de retourner les pions Joueur_Courant : joueur pour lequel on analyse le placement
Sorties :	Autorisation du choix de l'emplacement

Fonctionnement :

Autorisation_Choix = NOK
Vérification si la case choisie (X, Y) sur le damier est libre
    Si oui :
        - on regarde dans toutes les directions possibles si le pion placé entoure des pions adverses
        - les pions adverses entourés sont marqués comme entourés
        - Autorisation_Choix = OK

Module :	**coups_possible**
Rôle :	Fonction permettant de déterminer les coups possibles du joueur courant
Entrées :	Damier : damier de jeu sur lequel on travaille possible : coups possibles X et Y du damier Joueur_Courant : joueur pour lequel on analyse le placement
Sorties :	nombre de coups possibles.

Fonctionnement :

balayage du damier en X et en Y : (pour déterminer si le joueur courant peut poser un pion)
    si **retourne (**damier, X, Y, Joueur_Courant; flip = NOK**)** = OK
        possible_X (coup) = X
        possible_Y (coup) = Y
        nombre de coups possibles = nombre de coups possibles + 1
    fin_Si
fin de balayage

Module : **evaluation_damier**

Rôle : Fonction permettant d'évaluer un damier afin de déterminer la situation du jeu en lui donnant une note tel que :
- 1 < note < maxNote : jeu favorable aux Noirs
- - maxNote < Note < -1 jeu défavorable aux Noirs, donc favorable aux Blancs

Principe :

Sont attribuées des lettres à chaque case du damier :

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
1		C	A	B	B	A	C	
2	C	X	F	G	G	F	X	C
3	A	F	D	E	E	D	F	A
4	B	G	E			E	G	B
5	B	G	E			E	G	B
6	A	F	D	E	E	D	F	A
7	C	X	F	G	G	F	X	C
8		C	A	B	B	A	C	

On attribue une note en fonction du type de la case. La note sera attribuée de façon décroissante dans l'ordre suivant pour le joueur :
- les coins
- puis les cases A D B E G F C X

Puis on procède de façon inverse pour déterminer la note de l'adversaire.

Entrées : Damier : damier de jeu à évaluer

Sorties : Note du damier.

Fonctionnement :

Tableau de « score d'une case jouée par un joueur Noir » : TAB_CASE_NOIR

0	0	0	0	0	0	0	0	0
0	900	-220	90	70	70	90	-220	900	0
0	-220	-200	-20	0	0	-20	-200	-220	0
0	90	-30	30	40	20	40	-30	90	0
0	78	0	40	0	0	20	0	78	0
0	78	0	20	0	0	40	0	78	0
0	90	-30	40	20	40	30	-30	90	0
0	-220	-200	-20	0	0	-20	-200	-220	0
0	900	-220	90	70	70	90	-220	900	0
0	0	0	0	0	0	0	0	0	0

Tableau de « score d'une case jouée par un joueur Blanc » : TAB_CASE_BLANC

0	0	0	0	0	0	0	0	0	0
0	-2000	-90	-20	-15	-15	-20	90	-2000	0
0	80	200	10	30	30	10	100	90	0
0	-20	10	0	0	0	0	10	-20	0
0	-15	30	0	0	0	0	30	-15	0
0	-15	30	0	0	0	0	30	-15	0
0	-20	10	0	0	0	0	10	-20	0
0	80	200	10	30	30	10	100	90	0
0	-2000	-90	-20	-15	-15	-20	90	-2000	0
0	0	0	0	0	0	0	0	0	0

Note = 0

Détermination de la fin de jeu :
coupNoir = **coup_possibles (**damier, listeNoir, NOIR**)**
coupBlanc = **coup_possibles (**damier, listeBlanc, BLANC**)**

Si coupsNoir et coupBlanc = 0
- On détermine le gagnant en balayant le damier et en comptant les pions noirs et blancs.
- Si Blancs gagnants : Note du damier = -maxnote
- Si noirs gagnants : Note du damier = maxNote
- Si égalité : Note du damier = 0

Sinon
- calcul de la note du damier :
- Balayage en X et Y des tableaux de score d'une case jouée par un joueur Noir et d'un joueur Blanc
  - Si joueur_Courant = NOIR
    - Note du damier = Note du damier + TAB_CASE_NOIR (X,Y)+Coef de mobilité
  - Sinon
    - Note du damier = Note du damier + TAB_CASE_BLANC (X,Y) + coef de mobilité
- Fin balayage.

Module : **ordinateur_débutant**

Rôle : Fonction permettant de déterminer le coup de l'ordinateur pour le niveau initiation.

Principe :

Sont attribuées des lettres à chaque case du damier :

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
1			B	B	B	B		
2			C	C	C	C		
3	B	C	A	A	A	A	C	B
4	B	C	A			A	C	B
5	B	C	A			A	C	B
6	B	C	A	A	A	A	C	B
7			C	C	C	C		
8			B	B	B	B		

L'ordinateur ordonnera ses mouvements selon le type de la case. Ainsi l'ordre des mouvements sera :
- les coins
- puis les cases A B C
- et en dernier lieu, la première case vide

Entrées : CouleurOrdinateur.

Sorties : -

Fonctionnement :

CaseTrouvée = NOK

Priorité de jeu donnée aux 4 coins [(X,Y) = (1,1) - (1,8) - (8,8) - (8,1)]
Si **retourne (**damier, X, Y, CouleurOrdinateur; flip = OK**)** = OK
CaseTrouvée = OK
CoupX = X et CoupY= Y

Priorité 2 de jeu donnée aux cases de type A :
balayage en X et Y (à partir de X =3 et Y = 3)
Si **retourne (**damier, X, Y, CouleurOrdinateur; flip = OK**)** = OK
CaseTrouvée = OK
CoupX = X et CoupY= Y
Fin balayage

Priorité 3 de jeu donnée aux cases de type B :
balayage en X et Y (à partir de Y = 3)
Si **retourne (damier, 1, Y, CouleurOrdinateur; flip = OK)** = OK
CaseTrouvée = OK
CoupX = 1 et CoupY= Y
Fin balayage

Reprise du même type d'algo pour les 3 autres bords comportant des case B.

Priorité 4 de jeu donnée aux cases de type C :

Même type d'algorithme que précédemment.

Cas des autres cases : prise de la première case disponible.

Balayage en X et en Y

Si **retourne (**damier, X, Y, CouleurOrdinateur; flip = OK**)** = OK

CaseTrouvée = OK

CoupX = X et CoupY= Y

Si CaseTrouvée = OK

Affichage du coup joué : **jouer_coup (** X, Y, couleurOrdinateur**)**

Prise en compte du coup joué dans les statistiques : Statistique_Joueur_coup_X = X

Statistique_Joueur_coup_Y = Y

Sinon

Affichage fenêtre d'information : le joueur doit passer son tour

Module : **ordinateur_intermédiaire**

Rôle : Fonction permettant de déterminer le coup de l'ordinateur pour le niveau débutant.

Principe :

Sont attribuées des lettres à chaque case du damier :

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
1		C	A	B	B	A	C	
2	C	X	F	G	G	F	X	C
3	A	F	D	E	E	D	F	A
4	B	G	E			E	G	B
5	B	G	E			E	G	B
6	A	F	D	E	E	D	F	A
7	C	X	F	G	G	F	X	C
8		C	A	B	B	A	C	

L'ordinateur ordonnera ses mouvements selon le type de la case. Ainsi l'ordre des mouvements sera :
- les coins
- puis les cases A D B E G F C

Entrées : CouleurOrdinateur.

Sorties : -

Fonctionnement :

Réalisation d'un tableau priorisant les coups par rapport au damier définie plus haut (voir principe)
OrdreCoup (NB_COUPS) = (1,1) , (1,8) , ..., (7,7)
NB_COUPS = 60

Nombre de cases par type de case :
TypeCase(9) = { 4, 8, 4, 8, 8, 8, 8, 4, 8, 4} (coins, A, D, B, E, G, F, C, X)

Dans l'ordre des mouvements on détermine le coup que va jouer l'ordinateur :
Pour chaque COUPS possible de 1 à 60
Si **retourne (**damier, OrdreCoup (COUPS).x , OrdreCoup (COUPS).y CouleurOrdinateur; flip = OK**)** = OK
CoupX = OrdreCoup (COUPS).x
CoupY= OrdreCoup (COUPS).y

Balayage en X et Y du damier pour déterminer le score du coup
Si damier (X, Y) = couleurOrdinateur alors score(COUPS) = SCORE (COUP+1)
Fin du balayage
Fin pour

Détermination du meilleur coup dans chaque type de case possible :
Pour i de 1 à 8 et CaseTrouvée = OK
maxscore = 0
Pour j = 0 à typeCase (i)
Si score (j) = maxscore
X = OrdreCoups(j).x
Y = OrdreCoups(j).y
maxscore = score (j)

caseTrouvée = OK

Si caseTrouvée = OK
retournement des pions possible : **retourne (**Damier, X, Y, Couleurordinateur, flip = OK**)**
Affichage du coup joué : **Jouer_coup (**X, Y, CouleurOrdinateur**)**

Prise en compte du coup joué dans les statistiques : Statistique_Joueur_coup_X = X
Statistique_Joueur_coup_Y = Y

Module :	**alpha_beta**
Rôle :	Fonction permettant déterminer le meilleur coup de l'ordinateur en cherchant en profondeur en garantissant d'avoir le Pième coup à venir (P=profondeur) une note égale à l'un des coups joués à la profondeur P NOTA : C'est une fonction récursive
Entrées :	Damier Profondeur (de la recherche) Joueur X, Y : coordonnées du coup joué par le joueur Score note : note du coup qui : changement d'affectation des joueurs : utilisé pour la génération de l'arbre
Sorties :	-

Fonctionnement :

Appel = Appel + 1

on détermine si un des deux joueurs peut jouer
NB_COUPS = **coups_possible (**damier, liste, Joueur**)**
Si NB_COUPS = 0
    NB_COUPS = **coups_possible (**damier, liste, joueurAdverse**)**
    Si NB_COUPS = 0
        joueur bloqué : score = **evaluation_damier (**damier**)**

Si joueur = NOIR
    score = -MAX_NOTE
sinon
    score = MAX_NOTE

Affectation d'une profondeur de moins pour la recherche (profondeur = profondeur - 1)

Etude de tous les coups que le joueur peut jouer :
Balayage jusqu'à NB_COUPS
    Nouveau_Damier = Damier (copie du damier vers un damier tampon)
    retourne (Nouveau_Damier, liste.X(i), Liste.Y(i), Joueur, flip = OK**)** (on joue le coup)
    Nouveau_Damier (liste.X(i), liste.Y (i)) = joueur ;

    Si profondeur <> 0
        alpha_beta (Nouveau Damier, profondeur, Autre_Joueur, coup_x, coup_y, score, note=score, qui = joueur**)**
        (appel récursif de la fonction : on est ici à un noeud de l'arbre**)**
        le_score = **evaluation_damier(**Nouveau_Damier**)**
        (on est à une feuille de l'arbre de recherche)

    Si joueur = NOIR Procédure MAX en marche :
        Si (qui = BLANC et le_score >= note
            i = NB_COUPS
            score = MAX_NOTE
        sinon
            si le_score >= score
                score = le_score
                coup_X = liste.X (i)
                coup_Y = liste.Y (i)
                si score = MAX_NOTE
                    i = NB_COUPS

    Sinon (Procédure MIN en marche)
        Si (qui = NOIR et le_score <= note
            i = NB_COUPS
            score = - MAX_NOTE

```
sinon
        si le_score <= score
                score = le_score
                coup_X = liste.X (i)
                coup_Y = liste.Y (i)
                si score = - MAX_NOTE
                        i = NB_COUPS
```

Module :	**ordinateur_expert**
Rôle :	Fonction permettant déterminer le coup de l'ordinateur pour les hauts niveaux
Entrées :	CouleurOrdinateur couleurJoueur profondeur (de la recherche)
Sorties :	-

Fonctionnement :

Affichage du curseur « montre » durant la réflexion de l'ordinateur

recherche du meilleur coup avec la méthode alpha-Beta :
alpha-beta (Damier, profondeur, CouleurOrdinateur, X, Y, ScoreCoup, note = 0, qui = « »)

retourne (damier, X, Y, CouleurOrdinateur; flip = OK**)**

Affichage du coup joué **: jouer_coup (**X, Y, CouleurOrdinateur**)**

Prise en compte du coup joué dans les statistiques : Statistique_Joueur_coup_X = X
Statistique_Joueur_coup_Y = Y

Module :	**statistique**
Rôle :	Fonction permettant de calculer les statistiques des différents joueurs et de réaliser l'affichage vers les zones adéquates
Entrées :	Joueur_Courant : couleur du joueur demandant de recalculer les statistiques
Sorties :	-

Fonctionnement :

Définition du score de chaque joueur : comptage des pions : **compte_pions ()**

Définition de la mobilité des joueurs : détermination du nombre de coups possible de chaque joueurs (joueur courant et joueur adverse):
Statistique_mobilité_Joueur_Courant = **coups_possibles (**Damier, liste, Joueur_Courant**)**
Statistique_mobilité_Joueur_Adverse = **coups_possibles (**Damier, liste, Joueur_Adverse**)**

Définition de l'évaluation du coup du joueur courant :
Statistique_Evaluation_Joueur_Courant = **evaluation_damier ()**

Définition du nombre de feuilles analysées en recherche α-β
Statistique_Feuilles_Joueur_Courant = Appels
Appels = 0

Affichage des statistiques pour les joueurs (noir et blanc)
Affiche-statistique (NOIR**)**
Affiche_statistique (BLANC**)**

Module :	**statistique_affiche**
Rôle :	Fonction permettant d'afficher les statistiques
Entrées :	Joueur_Courant : couleur du joueur demandant de recalculer les statistiques
Sorties :	-

Fonctionnement :

Affichage du coup joué :
statistique_affiche_donnée ([Statistique_Joueur_coup_X + Statistique_Joueur_coup_Y])

Affichage de la mobilité
statistique_affiche_donnée (Statistique_mobilité_Joueur_Courant**)**

Affichage de l'évaluation
statistique_affiche_donnée (Statistique_evaluation_Joueur_Courant**)**

Affichage des données (profondeur + feuilles) uniquement si recherche α-β
statistique_affiche_donnée (Statistique_Feuilles_Joueur_Courant**)**
statistique_affiche_donnée (Profondeur**)**

Module :	**initiation**
Rôle :	Fonction permettant d'apprendre à jouer pour un joueur inexpérimenté
Entrées :	-
Sorties :	-

Fonctionnement :

initialisation des couleurs de pions pour le jeu :
demande du choix de la couleur dans une fenêtre de dialogue

Définition du type de jeu = initiation

Initialisation du jeu : **initialise_jeu ()**

Appel du jeu à deux joueurs : **jeu_2joueurs ()**

Module :	**affiche_coup**	
Rôle :	Fonction permettant d'afficher les coups possibles pour le joueur (niv. initiation)	
Entrées :	joueur :	joueur pour lequel on affiche les coups
	etat :	VRAI : coup visible - FAUX : coup non visible
Sorties :	-	

Fonctionnement :

détermination du nombre de coups possible pour le joueur :

appel de la fonction de calcul : **coup_possibles (**Damier, liste, Joueur**)**

affichage de points noir ou blancs en fonction de la couleur du joueur et des coups possibles.

Module :	**visualiser**
Rôle :	Fonction permettant de visualiser une partie sauvegardée
Entrées :	-
Sorties :	-

Fonctionnement :

initialisation de l'environnement de jeu : **initialise_ecran_jeu ()**

initialisation du jeu : remise à zéro ou initialisation des variables
- Couleur_joueurCourant
- Couleur_ordinateur
- Type de jeu

Saisie du nom de la partie à visualiser :
Affichage d'une boite de dialogue et saisie du nom de la partie (nomPartie)
Affichage en aide en ligne : « chargement en cours »
Charge_partie (nomPartie**)**

Si chargement = NOK alors fin module
sinon
affichage du nom des joueurs
initialisation de début de jeu : **statistique (**NOIR**)**

Tant qu'on ne quitte pas le jeu

Initialise_tableau_commande ()

rendre le bouton conseil inactif.

Attente click souris sur événements :
bouton coup précédent : **coup_precédent ()**
bouton coup suivant : **coup_suivant ()**

bouton stop : définition de la fin du jeu : fin de jeu = OK

Jusqu'à fin du jeu = OK

Module :	**fin_jeu**
Rôle :	Fin de la partie : un joueur a gagné / arrêt de la partie.
Entrées :	-
Sorties :	-

Fonctionnement :

Détermination du joueur gagnant

Si pas match nul
Affichage gagnant dans une boite de dialogue : « le joueur X a gagné avec x points d'écart »
Sinon
Affichage dans une boite de dialogue(« match nul »)

Attente click sur bouton OK de la boite

Remplissage du damier avec la couleur du pion du gagnant. (ça fout en rogne !!)

Module :	**apropos**
Rôle :	Affichage des informations concernant les concepteurs du jeu OTHO (c'est nous !!).
Entrées :	-
Sorties :	-

Fonctionnement :

Chargement de l'image apropos.gif dans le répertoire « images »

Masquage du curseur de la souris

Attente de quelques secondes

Disparition de l'image

Module :	**regles**
Rôle :	Fonction permettant de gérer les écrans de règles du jeu.
Entrées :	-
Sorties :	-

Fonctionnement :

gestion du curseur de la souris

Page_Courante = 1

tant que pas click sur le bouton « sortir »

Affichage de la page courante des règles du jeu : **regle_affiche (**Page_Courante**)**

Affichage du n° de la page courante : Page_Courante + « / 10 »

Affichage des boutons : **regle_affiche_bouton ()**

Rendre les boutons actifs et inactifs en fonction de la page
Si dernière page : bouton suivant inactif
Si première page, bouton précédent inactif.

Si click sur bouton précédent et page_Courante > 1 :
Page_Courante = Page_Courante-1

Si click sur bouton suivante et page_Courante < 10 :
Page_Courante = Page_Courante+1

Fin tant que.

Module :	**regles_affiche_bouton**
Rôle :	Fonction permettant d'afficher les boutons pour les écrans « règles du jeu »
Entrées :	-
Sorties :	-

Fonctionnement :

Affichage des boutons Précédent - Sortir - et Suivant

Réaliser une gestion si possible appuyé / relâché au niveau graphisme.

Module :	**regles_affiche**
Rôle :	Fonction permettant d'afficher l'écran courant des règles du jeu.
Entrées :	PAGE : Page courante à afficher
Sorties :	-

Fonctionnement :

Chargement de l'image dans le répertoire « \images\regles[PAGE].gif »

Module :	**sauve_partie**
Rôle :	Fonction permettant de sauvegarder une partie
Entrées :	-
Sorties :	-

Fonctionnement :

NOTA : *pour cette version la numérotation des fichiers est automatique (non respect de la STB)*

Recherche du nom du fichier pour la sauvegarde en vérifiant si le fichier (nom oth+ n°) existe déjà
Si tous les numéros sont pris (30 000)
impossibilité d'enregistrer la partie. (cela parait peu probable d'atteindre ce nombre)
Affichage sous forme de boite de dialogue : **«** dépassement de la capacité » et sortie de la boîte par un bouton OK
Sinon
Création du fichier de sauvegarde
- type de partie
- nom des joueurs
- indices du jeu
- tous les coups de la partie

Enregistrement physique sur le disque dur dans le répertoire « data »
Affichage sous forme de boîte de dialogue (partie enregistrée sous le nom othxxxx.dat) et sortie de la boite par un bouton OK

Module :	**charge_partie**
Rôle :	Fonction permettant de charger une partie
Entrées :	Nom du fichier contenant les données de la partie
Sorties :	Résultat du chargement (OK - NOK)

Fonctionnement :

Ouverture du fichier (\data\Nom du fichier.dat)

Si impossible : Affichage par boîte de dialogue : « Impossible d'ouvrir le fichier »
Sortie de la boîte par click sur bouton OK
Résultat du chargement = NOK

Sinon

positionnement en début de fichier
Montage en mémoire des éléments enregistrés :
- type de partie
- nom des joueurs
- indices du jeu
- tous les coups de la partie

Module :	**statistique_affiche_donnée**
Rôle :	Fonction de bas niveau permettant d'afficher en fonction de son entrée une donnée de statistique en cours de jeu sur l'écran (prise en compte des coordonnées à réaliser)
Entrées :	Donnée à afficher.
Sorties :	-

Module :	**high_score**
Rôle :	Fonction permettant de gérer les high-scores
Entrées :	-
Sorties :	-

Fonctionnement :

deltaScore = statistique_NOIR.score - statistique_BLANC.score

si deltaScore > 0 : gagnant = NOIR
si deltaScore < 0 : gagnant = BLANC
si deltaScore = 0 :egalité : pas de traitement

Détermination du niveau de jeu : jeu humain/humain ou humain/ordinateur

Si l'ordinateur est vainqueur : pas de traitement

Comparaison de deltaScore avec les autres scores des highscores
Si OK
insertion du score dans la liste des 5 meilleurs scores
Affectation du rang du gagnant
NomJoueur
score des noirs - score des blancs
date
décalage des autres scores
sauve_score ()
affiche_high_score (ECRAN_JEU**)**

Module :	**affiche_high_score**
Rôle :	Fonction permettant d'afficher le tableau complet des high-scores
Entrées :	NomEcran : permet de savoir si l'affichage du tableau est commandée de l'écran de jeu ou du menu principal.
Sorties :	-

Fonctionnement :

Chargement de l'image « \images\otho.gif »

Affichage du tableau des high-scores vide

pour chaque type de donnée à afficher :
    NomJoueur
    score des noirs - score des blancs
    date
    Statistique_affiche_données (données**)** en fonction des coordonnées.
fin pour

Si appui sur OK et positionnement NomEcran = Ecran de jeu
    effacement de l'image
    Chargement de l'image « \images\otho.gif »
    Initialise_Jeu ()
fin si

Module :	**sauve_score**
Rôle :	Fonction permettant de sauvegarder sur un fichier les scores du jeu
Entrées :	-
Sorties :	-

Fonctionnement :

Si le fichier « data\scores.dat » n'existe pas :le créer
    Si problème à la création : affichage d'un message d'erreur.
Sinon
    écrire dans le fichier la ligne correspondante au high-score réalisé

Module :	**charge_score**
Rôle :	Fonction permettant de monter en mémoire le fichier High-score
Entrées :	-
Sorties :	résultat du chargement (OK - NOK)

Fonctionnement :

Ouverture du fichier « data\scores.dat »

S'il n'existe pas : Chargement = NOK
Sinon
    positionnement en début de fichier
    chargement des données en mémoire
    fermer le fichier
    Chargement = OK

Module :	**ouverture**
Rôle :	Fonction permettant d'accéder à la bibliothèque des ouvertures.
Entrées :	-
Sorties :	-

Fonctionnement :

Chargement de l'image « \images\otho.gif »

Affichage de la boîte de dialogue de choix de l'ouverture
Affichage de la boîte d'aide en ligne

Préparation de la bibliothèque d'ouvertures : retour = **prepare_ouverture ()**

Si retour = OK alors
    affichage en aide en ligne : « Faites votre choix »
    retour = **selection_ouverture ()**
    si retour = OK alors
        Affichage en aide en ligne: « sélection de l'ouverture : », nomOuverture
        ouverture_visualise (retour**)**

Module :	**prepare_ouverture**
Rôle :	Fonction permettant de vérifier l'existence des fichiers de données et de créer éventuellement la bibliothèque.
Entrées :	-
Sorties :	-

Fonctionnement :

Vérification de l'existence du fichier « \data\ouvertur.dat » (format binaire)

Si le fichier n'existe pas : le créer à partir du fichier « data\ouvertur.txt »
    Si pas trouvé sur disque :
        Affichage en aide en ligne : « le fichier des ouvertures n'existe pas !! »
    NOK
    Si trouvé :
        transformation du fichier .txt en .dat
        OK
        Si problème :
            Affichage en aide en ligne : « il y a un léger problème ... »
            NOK
    Fin Si
Fin Si

Module :	**ouverture_visualise**
Rôle :	Fonction permettant de visualiser l'ouverture sélectionnée.
Entrées :	selection : indice dans la table ouverture
Sorties :	-

Fonctionnement :

Le but est de déterminer le nombre de coups de l'ouverture et de préparer la table des coups à jouer
nombrecoups = 0

Reconnaissance des caractères majuscule - minuscule pour le coup à jouer
Si majuscule :
couleurJoueur = Noir
reconnaissance et enregistrement du coup en X, Y
Calcul au fur et à mesure du nombre de coup
Si minuscule : couleurJoueur = Blanc et reconnaissance du coup en X, Y
couleurJoueur = Blanc
reconnaissance et enregistrement du coup en X, Y
Calcul au fur et à mesure du nombre de coup
Affichage en aide en ligne : « l'ouverture comporte », nombreCoups » « coups. »

initialise_ecran_jeu ()

initialisation des variables courantes dont JoueurCourant = Noir

initialise_Jeu ()

Affichage des statistique de début de jeu : **statistique (**NOIR**)**

Tant qu'on ne quitte pas le jeu

Initialise_tableau_commande ()

rendre le bouton conseil inactif.

Attente click souris sur événements :

bouton coup précédent : **coup_precédent ()**
Si nombrecoups <> 0
nombrecoup = nombrecoup-1
Affichage en aide en ligne : « Coup courant de l'ouverture : »,nombrecoup/totalcoup
Sinon
Affichage en aide en ligne : « Vous êtes au début de l'ouverture, ducon !!! »

bouton coup suivant :
Vérification de coup possible (pas arrivé à la fin des coups de l'ouverture)
Si oui : lecture du coup à jouer : CaseX, CaseY
retourne (damier, caseX, caseY, CouleurJoueurCourant, flip = OK**)**
jouer_coup (caseX, caseY, CouleurJoueurCourant**)**
calcul des statistiques pour le coup en CaseX CaseY
passage au coup suivant : CoupCourant = CoupCourant +1
statistique (CouleurJoueurCourant)
affiche_pions ()
memorise_jeu (CouleurJoueurCourant**)**
change_joueur ()
Affichage en aide en ligne : « Coup courant de l'ouverture : »
,nombrecoup/totalcoup
Si non : Affichage en aide en ligne : « dernier coup de l'ouverture atteint !! »
Affichage des coups possibles :**affiche_coup (**CouleurJoueurCourant)

bouton stop : définition de la fin du jeu : fin de jeu = OK

Jusqu'à fin du jeu = OK

Module :	**selection_ouverture**
Rôle :	Gestion de la sélection de l'ouverture avec la souris
Entrées :	-
Sorties :	NomOuverture

Fonctionnement :

préparation des pages d'ouvertures :
    positionnement en début de fichier
    balayage numérique (i)
        Page (i) = 28 lignes du fichier
        i = i+1
    fin de balayage

PageCourante = 1

tant que pas clic sur Annuler ou OK

    affichage de la page courante si nouvelle page demandée (ou début d'opération)

    Click sur bouton >>
        pageCourante = PageCourante + 1

    Click sur bouton <<
        pageCourante = PageCourante - 1

    Click sur bouton + :
        Si dernière ligne de la page : rester à l'ouverture courante
        Sinon
            Sélection de la ligne suivante et passage en surbrillance du nom de l'ouverture
            Sélection = OuvertureCourante
        FinSi

    Click sur bouton - :
        Si première ligne de la page : rester à l'ouverture courante
        Sinon
            sélection de la ligne précédente et passage en surbrillance du nom de l'ouverture
            Selection = OuvertureCourante
        FinSi

    Click sur bouton OK :
        NomOuverture = Selection

    Click sur le bouton Annuler:
        NomOuverture = « »

Fin tant que

5. CODAGE :

5.1 Règles utilisées pour le codage

5.1.1 Cartouche

Le type de cartouche ci-dessous est utilisé pour tout module composant OTHO.

```
/*************************************************************************/
/* Nom : NOM_DU_MODULE                                                   */
/*                                                                       */
/* Role : Définition rapide du rôle du module dans le logiciel           */
/*                                                                       */
/* Entree : Paramètres d'entrées du module                               */
/* Sortie : Paramètres de sortie du module                               */
/*                                                                       */
/* ===================================================================== */
/* Nom | Date       | Description                                        */
/* ----+------------+--------------------------------------------------- */
/* xxx | xx/xx/1997 | Creation de la fonction.                           */
/*     |            |                                                    */
/* xxx | xx/xx/1997 | Ajout de la gestion des xxxxxxx                    */
/*     |            |                                                    */
/*     |            |                                                    */
/*     |            |                                                    */
/*************************************************************************/
```

5.1.2 Structure du code :

Notre code est structuré de la manière suivante :

CARTOUCHE

INCLUDES

DECLARATIONS

CORPS DU MODULE

5.1.3 Lisibilité

Un effort est porté sur la lisibilité du code afin d'optimiser la maintenance et l'évolution de fonction :
Pour atteindre cet objectif, le code :
- sera indenté pour chaque boucle, branchement If, case, ...
- aura un maximum de commentaires pour sa compréhensibilité
- sera aéré pour sa lisibilité
- sera modulaire pour accroître la facilité de dépannage (réalisation au maximum de petits modules)

5.1.4 Règles de nommage des variables :

Les noms des variables utilisées devront être le plus clairs possible sur leur utilité et sur leur représentation.

Exemple :
joueurCourant
couleurOrdinateur

Exception faite des variables locales de boucle qui, par habitude, s'appellent i, j, ..

Les noms des variables globales seront précédées par un G pour indiquer leur statut particulier de variables globales.

Exemples :
G_board
G_statistique [G_joueurCourant].coupX

5.2 Graphe d'appel des fonctions codées

Ce type de graphe nous permet de visualiser graphiquement les appels entre fonctions et ainsi de connaître la structure du logiciel. C'est un outil pratique pour l'analyse de l'impact des évolutions, mais aussi pour la réalisation de ces évolutions de fonctionnalité.

Graphe des appels : voir annexe n° 2.

5.3 Fichiers sources et leur contenu :

Quatre grands types de répertoires ont été créés ou utilisés pour le codage d'OTHO :

- include : pour regrouper tous les fichiers « include » (entête des fichiers sources)
- lib pour regrouper tous les fichiers de la librairie utilisée
- src : pour regrouper tous les fichiers sources.
- tc : c'est le répertoire contenant la version 3.0 de Borland Turbo C++

5.3.1 Sources - fichiers « .c »

Contenu du répertoire SRC :

2joueur.c	14923	21/05/1997	23:08:12
algorith.c	38368	21/05/1997	22:53:44
apropos.c	1959	27/04/1997	14:35:50
capture.c	3754	25/05/1994	11:12:18
dialogue.c	45081	21/05/1997	23:01:08
ecran1.c	2798	27/04/1997	14:36:36
fichier.c	5588	21/05/1997	22:09:56
gifsave.c	31590	25/05/1994	09:59:50
initiat.c	3635	21/05/1997	22:04:28
jouer.c	42531	21/05/1997	23:12:46
keyboard.c	1321	15/12/1996	16:36:28
materiel.c	6401	27/04/1997	14:36:48
memoire.c	5702	27/04/1997	14:37:00
menupal.c	9872	25/05/1994	10:59:08
ordinat.c	6903	21/05/1997	22:06:04
otho.c	3448	25/05/1994	09:53:32
ouvert.c	28290	21/05/1997	23:04:14
regles.c	13409	19/05/1997	18:13:20
scores.c	13643	21/05/1997	21:36:36
services.c	3968	14/01/1997	00:21:42
souris.c	9183	25/05/1994	10:59:30
statist.c	8637	21/05/1997	23:29:28
variable.c	2920	18/05/1997	16:10:06
video.c	9400	25/05/1994	11:14:52
visual.c	4154	19/05/1997	18:22:08

Un tableau concernant la taille de l'application en termes de nombres de lignes (code + commentaire) ainsi qu'en terme d'octets est cité en annexe n°3. Ce tableau a pour mérite de représenter l'effort réalisé en terme de codage.

Contenu des fichiers :

Nom du fichier	Module(s) contenu(s)		Nom du fichier	Module(s) contenu(s)

2joueur.c	jeu_2joueur fin_jeu
algorith.c	retourne coups_possibles evaluation_damier ordinateur_debutant ordinateur_intermédiaire ordinateur_expert min_max alpha_beta
apropos.c	apropos
capture.c	capture_ecran
ecran1.c	ecran_presentation
capture.c	capture_ecran
dialogue.c	dialogue_boite dialogue_cadre dialogue_zone_saisie dialogue_bouton_relache dialogue_bouton_appuye dialogue_OK dialogue dialogue_type_jeu dialogue_gere_choix dialogue_etat_choix dialogue_aide_en_ligne dialogue_choix_couleur dialogue_OK_couleur dialogue_passe_tour dialogue_saisie dialogue_gestion_clavier
fichier.c	sauve_partie charge_partie
gifsave.c	*voir NOTA*
initiat.c	initiation affiche_coup
jouer.c	jouer initialise_ecran_jeu initialise_objet_graphique initialise_jeu initialise_tableau_commande compte_pions raz_zone_score affiche_nombre affiche_pions choix_joueur change_joueur jouer_coup coup_precedent coup_suivant memorise_jeu conseil_coup jeu_normal
keyboard.c	getkey
materiel.c	materiel

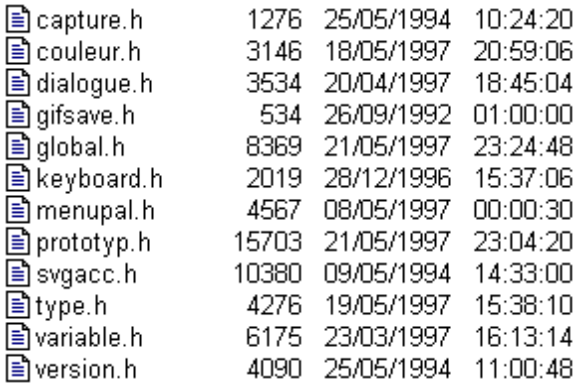
memoire.c	allocation_memoire libere_memoire
menupal.c	menu_principal affiche_choix
ordinat.c	ordinateur_vs_ordinateur
otho.c	main
ouvert.c	ouverture ouverture_visualise creation_bibliotheque_ouverture prepare_ouverture affiche_page_ouverture ouverture_gestion_souris selection_ouverture
regles.c	regles regles_affiche_bouton regles_affiche regles_bouton_relache regles_bouton_appuye
score.c	high_score affiche_high_score sauve_score charge_score
services.c	attente erreur_application
souris.c	change_curseur mouse_attente mouse_attente_relache mouse_zone mouse_visible
statist.c	statistique statistique_affiche statistique_affiche_donnee
variable.c	initialise_variables
video.c	depart_video video_on video_off loadpal charge_image
visual.c	visualiser

NOTA : *Le fichier gifsave.c est un fichier issu de la librairie utilisée : récupération de fonction pour les captures d'écran. Il n'y a donc eu aucune intervention ou modification de code*

5.3.2 Include - fichiers .h :

Contenu du répertoire Include :

Les fichiers INCLUDE (.h) sont les fichiers d'en-tête des fichiers sources. Ils contiennent la définition des variables et éventuellement le prototypage des fonctions utilisés.



capture.h	1276	25/05/1994	10:24:20
couleur.h	3146	18/05/1997	20:59:06
dialogue.h	3534	20/04/1997	18:45:04
gifsave.h	534	26/09/1992	01:00:00
global.h	8369	21/05/1997	23:24:48
keyboard.h	2019	28/12/1996	15:37:06
menupal.h	4567	08/05/1997	00:00:30
prototyp.h	15703	21/05/1997	23:04:20
svgacc.h	10380	09/05/1994	14:33:00
type.h	4276	19/05/1997	15:38:10
variable.h	6175	23/03/1997	16:13:14
version.h	4090	25/05/1994	11:00:48

Contenu des fichiers de type .h :

```
/*************************************************************************/
/* Nom : fichier capture.h                                               */
/*                                                                       */
/* Role : Definition des variables utilisees pour la capture             */
/*************************************************************************/

/*************************************************************************/
/* Nom : fichier couleur.h                                               */
/*                                                                       */
/* Role : Definition des couleurs des differents ecrans                  */
/*************************************************************************/

/*************************************************************************/
/* Nom : fichier dialogue.h                                              */
/*                                                                       */
/* Role : Declaration des variables utilisees dans dialogue.c            */
/*************************************************************************/


Fichier gifsave.h :          RECUPERE
      declaration des variables utilisées dans gifsave.c


/*************************************************************************/
/* Nom : fichier global.h                                                */
/*                                                                       */
/* Role : Declaration des variables globales                             */
/*************************************************************************/

/*************************************************************************/
/* Nom : fichier keyboard.h                                              */
/*                                                                       */
/* Role : Definition des touches du clavier                              */
/*************************************************************************/

/*************************************************************************/
/* Nom : fichier menupal.h                                               */
/*                                                                       */
/* Role : Declaration des variables utilisees dans menupal.c             */
/*************************************************************************/

/*************************************************************************/
/* Nom : Fichier prototyp.h                                              */
/*                                                                       */
/* Role : Prototypage des fonctions utilisees                            */
/*************************************************************************/
```

Fichier SVGACC.h : **RECUPERE**

```
      /* SVGACC Include File for MS C/C++/QuickC and Borland C/C++
       * Copyright 1994 by Stephen L. Balkum and Daniel A. Sill
       * Zephyr Software P.O. Box 7704, Austin, Texas  78713-7704
       */

/*************************************************************************/
/* Nom : fichier type.h                                                  */
/*                                                                       */
/* Role : Definition des types                                           */
/*************************************************************************/

/*************************************************************************/
/* Nom : fichier variable.h                                              */
/*                                                                       */
/* Role : Declaration des variables utilisees dans variable.c            */
/*        Dessin des curseurs souris                                     */
/*************************************************************************/

/*************************************************************************/
/* Nom : fichier version.h                                               */
/*                                                                       */
/* Role : Tracabilite de toutes les evolutions du jeu otho               */
/*************************************************************************/
```

5.3.3 Lib : librairie

C'est le répertoire contenant toutes les fonctions de la librairie graphique que nous utilisons.

readme.1st	7743	09/05/1994	14:33:00
svgadem1.c	25978	09/05/1994	14:33:00
svgadem2.c	31124	09/05/1994	14:33:00
svgademo.c	14780	09/05/1994	14:33:00
history.cc	3830	09/05/1994	14:33:00
file_id.diz	360	09/05/1994	14:33:00
svgademo.exe	101686	09/05/1994	14:33:00
dragon.fnt	4098	09/05/1994	14:33:00
order.frm	8451	09/05/1994	14:33:00
uttower.gif	60849	09/05/1994	14:33:00
svgacc.h	10380	09/05/1994	14:33:00
svgademo.h	7423	09/05/1994	14:33:00
svgacc.lib	70847	09/05/1994	14:33:00
packing.lst	1231	09/05/1994	14:33:00
desc.sdi	68	09/05/1994	14:33:00
svgacc21.txt	299680	09/05/1994	14:33:00

6. MATRICE DE TRACABILITE DES EXIGENCES :

SERVICE (CC)	EXIGENCE (STB)	FONCTIONS (DCP)	MODULES (DCD)
INITIALISER	[INI.1]	[ACC.1] [ACC.2]	[materiel] [allocation_memoire] [ecran_presentation]
	[INI.2]	[MEN.1] [MEN.2] [MEN.3] [REG.1] [REG.2] [REG.3] [PRO.1] [PRO.2] [PRO.3]	[menu_principal] [menu_principal] [menu_principal] [regles] [regle_affiche_bouton] [regles] [regle_affiche] [apropos] [apropos] [apropos]
	[INI.3]	[PAR.1] [PAR.2] [PAR.3]	[jouer] [initialise_ecran_jeu] - [choix_joueur] [jeu_normal] - [initiation] [jeu_normal]
JOUER UN COUP0	[JOU.1]	[GRI.1] [GRI.2]	[jeu_2joueur] [statistique] - [initialise_tableau_commande]
	[JOU.2]	[PLA.1] [PLA.2] [PLA.3]	[jeu_2joueur] - [ordinateur_debutant] [ordinateur_intermédiaire] [ordinateur_expert] [ordinateur_debutant] [evaluation_damier] [alpha_beta] [coup_possible] [change_joueur]
	[JOU.3]	[PLA.4]	[jeu_2joueur]
	*	[PLA.5] [PLA.6] [PLA.7]	[jeu_2joueur] [conseil_coup] [jeu_2joueur]
	[JOU.4]	[VER.1] [VER.2]	[jeu_2joueur] [jouer_coup] [retourne] [retourne]
	[JOU.5]	[COU.1] [COU.2] [COU.3] [COU.4]	[jouer_coup] [retourne] [retourne] [statistique] [statistique_affiche]
	[JOU.6]	[COU.1] [COU.2] [COU.3] [COU.4]	[jouer_coup] [retourne] [retourne] [statistique] [statistique_affiche]
	[JOU.7]	[FIN.1]	[fin_jeu]
GERER LES SCORES	[SCO.1]	[CAL.1] [CAL.2]	 [compte_pions] [statistique]
	[SCO.2]	[VSC.1] [VSC.2] [PPE.1] [PPE.2] [PGA.1] [PGA.2] [PGH.1] [PGH.2]	[fin_jeu] [high_scores] [fin_jeu] [high_scores] [fin_jeu] [high_scores] [affiche_ecran_score] [fin_jeu] [high_scores] [sauve_score]
	[SCO.3]	[ATA.1]	[affiche_ecran_score]

SERVICE (CC)	EXIGENCE (STB)	FONCTIONS (DCP)	MODULES (DCD)
GERER LA BIBLIOTHEQUE	[BIB.1]	[CON.1] [LAN.1] [LAN.2] [SAU.1] [SAU.2]	 [jeu_2joueur] [sauve_partie] [sauve_partie]
	[BIB.2]	[CHO.1] [CHO.2] [CHO.3] [CHO.4] [CHO.5] [CHO.6] [CHO.7] [VIS.1] [VIS.2] [VIS.3]	[visualiser] [initialise_ecran_jeu] [visualiser] [charge_partie] [charge_partie] [visualiser] [charge_partie] [ouverture] [prepare_ouverture] [selection_ouverture] [prepare_ouverture] [visualiser] [initialise_ecran_jeu] [initialise_tableau_commande] [ouverture_visualise] [coup_precedent] [coup_suivant] [fin_jeu]
	[BIB.3]	[REP.1] [REP.2] [REP.3] [REP.4]	[jeu_normal] [charge_partie] [charge_partie] [charge_partie] [jeu_2joueur]
Exigences de développement	[DEV.1]	*Non développé dans le DCP*	*developpé dans le § 5.1*
	[DEV.2]	*Non développé dans le DCP*	*du fait de l'utilisation de Turbo C*
	[DEV.3]	*Non développé dans le DCP*	*développé dand le § 5.1*

Au vu de ce tableau, on peut conclure que certaines exigences n'ont pas été respectées. Par contre, des fonctions ont été ajoutées, comme par exemple, le combat entre ordinateurs (module [ordinateur_vs_ordinateur] qui n'apparaît pas ici) qui est le déroulement d'une partie OTHO contre lui-même (à des niveaux différents).