CAHIER DE CONCEPTION DÉTAILLÉ

Projet d'algorithmique 2016-2017

Version: CCD_Groupe04_RJA-OAK-ABE_V1.3
Auteur: Romain JACQUIEZ, Ouassim AKEBLI et Antoine BERENGUER

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

Description du document

Type	Version	Confidentialité
Cahier de conception détaillée	1.3	Usage externe

	Nom	Fonction	Date	Visa
Rédacteur	Romain JACQUIEZ Ouassim AKEBLI	Rédaction	04/01/17 au 07/01/17	
Vérificateur	Romain JACQUIEZ	Corrections	12/01/17	
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	AB	Rédaction de contenu
1.2	05/01/17	RJ	Rédaction de contenu
1.3	10/01/17	RJ	Rédaction de contenu

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

Les cahiers des charges, de recette, et de conception générale permettraient avec très peu de modifications d'adapter le projet pour n'importe quel langage de programmation. Dans notre cas, nous utilisons le langage C99 et ce cahier de conception détaillé permettra d'expliciter le fonctionnement des parties les plus complexes de notre programme.

2. DÉTAIL DES STRUCTURES DE DONNÉES

A) Étude détaillée

Toutes les coordonnées du tableau seront stockées dans des structures, ainsi cela simplifie le passage en paramètre dans les fonctions d'affichage et/ou de traitement ainsi que les retours.

Comme indiqué dans le cahier de conception générale, les informations des joueurs sont aussi stockées dans des structures.

Le plateau de jeu est géré par un tableau d'entiers de deux dimensions

Nous aurons aussi besoin d'un tableau de structures dans lequel chaque structure contient les coordonnées x et y des quartes coins qui forment la case de la grille.

B) Description en C commentée

```
#define MAX 5
typedef int Plateau[MAX][MAX]; //déclaration du type de la grille de Quixo

typedef struct infoJoueur //informations concernant un joueur
{
	char *nomJoueur; //facultatif
	char symboleJoueur; //Symbole utilisé pour représenter le joueur sur la grille
	int scoreJoueur; //Score du joueur sur les parties
	struct *infoJoueur joueurSuivant; //pointeur vers le joueur suivant
} infoJoueur, *pinfoJoueur;

typedef struct coord //structure représentant les coordonnées d'une case en mode console
{
	int x;
	int y;
} coord, *pcoord;

typedef struct case //structure représentant les coordonnées d'une case en mode graphique
{
	int x1 ;
	int y1 ;
	int x2 ;
	int y2 ;
	int x3 ;
	int y3 ;
	int x4 ;
	int y4 ;
}case ;

typedef case tabcase[MAX][MAX] ; //tableau qui contient les structs qui elles-même contiennent les coordonnées de chaque cases en mode graphique.

tabcase *ptabcase ; //pointeur vers un tableau 2D de structs
```

3. DESCRIPTION DES FONCTIONS

A) Description des fonctions

Pseudo-code pour des fonctions

```
Gagne_colonne(Grille)
Début
        variables a, colonne, valeur et somme
        pour colonne de 0 à MAX-1 : //On regarde toutes les colonnes
                pour a de 0 à MAX-1 : //Dans la colonne, on vérifie toutes les cases
                        si Grille[0][colonne] = Grille[a][colonne] != 0
                //Si la première case est égale à la case que l'on vérifie et est non nulle
                        alors :
                                somme <- somme + 1
                //On incrémente la somme de marqueurs identiques
                        sinon :
                                somme <- 0 //On réinitialise
                si somme = MAX
                //Si la somme de marqueurs requise pour gagner est atteinte
                alors :
                        valeur <- Grille[0][colonne];
                //On récupère la valeur du marqueur gagnant
                        retour : valeur //On retourne ce marqueur
                sinon:
                        retour : 0 //Pas de gagnant, on retourne 0
Fin

Decale_ligne_gauche(pGrille, int ligne, int indice)
Début
        variables échangeur, a
        pour a de indice à MAX-2; //parcours de la ligne de gauche à droite
                echangeur <- valeur(pGrille)[ligne][a]
                //valeur(pointeur) correspond à la valeur pointée par la variable "pointeur"
                valeur(pGrille)[ligne][a] <- valeur(pGrille)[ligne][a+1]
                valeur(pGrille)[ligne][a+1] <- echangeur
                //On échange chaque valeur avec celle de droite à l'aide d'un échangeur,
                résultant en un décalage de 1 vers la gauche
        retour : void
Fin
```

```
Decale_ligne_droite(pGrille, int ligne, int indice)
Début
	variables echangeur, a
	pour a de indice à 1 en décrémentant;
	//parcours de la ligne de droite à gauche
		echangeur <- valeur(pGrille)[ligne][a]
	//valeur(pointeur) correspond à la valeur pointée par la variable "pointeur"
		valeur(pGrille)[ligne][a] <- valeur(pGrille)[ligne][a-1]
		valeur(pGrille)[ligne][a-1] <- echangeur
	//On échange chaque valeur avec celle de gauche à l'aide d'un échangeur, résultant
	en un décalage de 1 vers la droite
	retour : void
Fin

Choix_haut(coord coordCase)
Début
	si coordCase.y = 0
		retour : NULL
	//on est déjà en haut de la grille, il n'est pas possible de choisir une case plus haute
	variable : pointeur vers coord "pCoordHaut"
	pCoordHaut <- Allocation mémoire de la taille d'une variable de type coord
	valeur(pCoordHaut).y <- 0
	valeur(pCoordHaut).x <- coordCase.x
	retour : pCoordHaut
Fin

init_coord_grille(tabcase *ptabcase)
Début
	variables : i,j
	//On commence par calculer les coins de la grille
	xMin <- ... //abscisse minimum de la grille
	yMin <- ... //ordonnée minimum de la grille
	xMax <- ... //abscisse maximum de la grille
	yMax <- ... //ordonnée maximum de la grille
	pour i de 0 à MAX-1 :
//On calcule maintenant les coordonnées des quatres points de chaque case de la grille
		pour j de 0 à MAX-1 :
			valeur(ptabcase)[i][j].x1 <- xMin + (j*((xMax-xMin)/5))
			valeur(ptabcase)[i][j].y1 <- yMax - ((i+1)*((yMax-yMin)/5))
			valeur(ptabcase)[i][j].x2 <- xMin + ((j+1)*((xMax-xMin)/5))
			valeur(ptabcase)[i][j].y2 <- yMax - ((i+1)*((yMax-yMin)/5))
			valeur(ptabcase)[i][j].x3 <- xMin + (j*((xMax-xMin)/5))
			valeur(ptabcase)[i][j].y3 <- yMax - (i*((yMax-yMin)/5))
			valeur(ptabcase)[i][j].x4 <- xMin + ((j+1)*((xMax-xMin)/5))
			valeur(ptabcase)[i][j].y4 <- yMax - (i*((yMax-yMin)/5))
Fin
```

```
detecte_case_clic (int xMin, int yMin, int xMax, int yMax, int abs, int ord)
Début
	si abs < xMin ou abs > xMax ou ord < yMin ou ord > yMax
		retour : NULL
	//Le clic a été fait en dehors de la grille. On ignore.
	variables : i, j, coordoX, coordoY
	variable : pointeur vers coord "pCoordClic"
	pCoordClic <- Allocation mémoire de la taille d'une variable de type coord
	//On crée la structure dont on retournera le pointeur
	xTraitement = abs - xMin
	xTraitement = xTraitement * (1000/(xMax-xMin))
	//On transforme les cordonnées en un nombre entre 0 et 1000 pour un traitement
	plus facile
	pour i de 1 à MAX :
	//A l'aide d'une boucle, on identifie dans quelle colonne le joueur a cliqué.
		si (xTraitement>=((1000/5)*(i-1)) et xTraitement<((1000/5)*(i)))
		{
			coordoX <- i-1;
		}
	valeur(pCoordClic).x <- coordoX
	//On stocke la colonne trouvée dans la structure
	yTraitement = ord - yMin
	//On répète l'opération avec la coordonnée de la ligne
	yTraitement = yTraitement * (1000/(yMax-yMin))
	pour i de 1 à MAX :
		si (yTraitement>=((1000/5)*(i-1)) et yTraitement<((1000/5)*(i)))
		{
			coordoY <- i-1;
		}
	valeur(pCoordClic).y <- coordoY
	retour : pCoordClic
Fin

croix (tabcase *ptabcase, coord CoordClic)
Début
	variable : offset
	offset <- 5
//Cette valeur permet de rendre la croix plus petite dans la case dans le calcul des
coordonnées.
//On calcule les coordonnées de la croix à partir du tableau de coordonnées et de l'offset
	x1Ligne <- valeur(ptabcase)[CoordClic.x][CoordClic.y].x1 + offset
	y1Ligne <- valeur(ptabcase)[CoordClic.x][CoordClic.y].y1 + offset
	x2Ligne <- valeur(ptabcase)[CoordClic.x][CoordClic.y].x2 - offset
	y2Ligne <- valeur(ptabcase)[CoordClic.x][CoordClic.y].y2 + offset
	x3Ligne <- valeur(ptabcase)[CoordClic.x][CoordClic.y].x3 + offset
	y3Ligne <- valeur(ptabcase)[CoordClic.x][CoordClic.y].y3 - offset
	x4Ligne <- valeur(ptabcase)[CoordClic.x][CoordClic.y].x4 - offset
	y4Ligne <- valeur(ptabcase)[CoordClic.x][CoordClic.y].y4 - offset
	//Avec les coordonnées, on trace deux lignes résultant en une croix.
```

Note: il faut également définir l'épaisseur du trait et la couleur.
ligne(x1Ligne,y1Ligne,x4Ligne,y4Ligne)
ligne(x2Ligne,y2Ligne,x3Ligne,y3Ligne)
retour : void
Fin

B) Erreurs et oublis