CAHIER DE CONCEPTION DÉTAILLÉE

Projet d'algorithmique 2016-2017

Version: CCD_Groupe04_GDE_MBO_PPA_V1.1
Auteur: ISEN Toulon

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	1.1	Usage externe

	Nom	Fonction	Date	Visa
Rédacteur	ISEN Toulon			
Vérificateur				
Approbateur				

Destinataire	Fonction	Organisme
Public		ISEN

Révisions du document

Version	Date	Rédacteur	Modifications
1.0	18/07/2016	FMC	Mise en forme
1.1	22/01/2017	GDE / MBO / PPA	Ajout de la documentation

Sommaire

1. INTRODUCTION.....6
2. DÉTAIL DES STRUCTURES DE DONNÉES.....6
A) Étude détaillée.....6
B) Description en C commentée.....7
3. DESCRIPTION DES FONCTIONS.....8
A) Description des fonctions.....8
B) Erreurs et oublis.....14

Index des illustrations

Illustration 1: Arbre d'appel des fonctions.....6

Index des tables

REFERENCES

Référence	Description	Nom
[1]		
[2]		

DEFINITIONS

Sans objet

ABBREVIATIONS

ISEN : Institut Supérieur de l'Electronique et du Numérique

SQL : Structured Query Language

1. INTRODUCTION

Dans cette partie nous allons analyser en détail le fonctionnement de notre programme en expliquant le choix des structures de données choisies pour le jeu du Quixo en fournissant une étude détaillée ainsi qu'une description en C de celles-ci.
Les fonctions présentes dans le programme vont être décrites en pseudo-code pour une meilleure compréhension. Les erreurs et oublis rencontrés lors de l'élaboration du projet seront décrites en dernière partie du document.

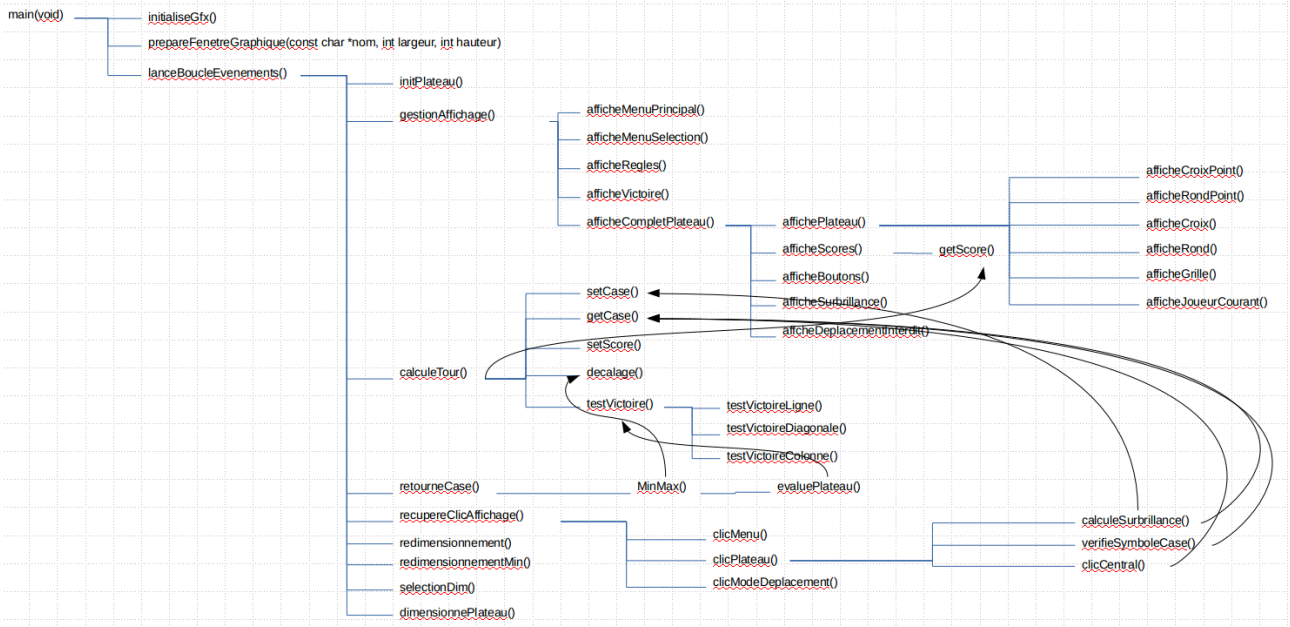

Illustration 1: Arbre d'appel des fonctions

2. DÉTAIL DES STRUCTURES DE DONNÉES

A) Étude détaillée

- Le plateau du jeu est un tableau d'entiers 2D de 6*6 cases, on ajoute une zone tampon au jeu en 5*5 afin de pouvoir permettre au joueur d'user d'une zone de placement.
  → Type PLATEAU
- Les symboles des cubes sur le plateau de jeu sont représentés par une énumération de la forme :
  - croix_bas (1)
  - rond_bas (2)
  - croix_gauche (3)
  - rond_gauche (4)
  - croix_haut (5)
  - rond_haut (6)
  - croix_droit (7)

  - rond_droit (8)
  - vide (9)
  - surbrillance (10)
  - tampon (11)

  Les nombres impairs sont destinés au croix et les nombres pairs sont destinés aux ronds.
- La constante NB_VICTOIRE est égale à 5. Elle représente le nombre de cube de même symbole qu'il faut aligner (verticalement, horizontalement ou diagonalement) pour remporter la partie.
- La constante TAILLE_PLATEAU est égale à 6. Elle représente la hauteur et la largeur de la grille.
- La constante TAILLE_FENETRE est égale à 800, qui indique la taille minimale que peut avoir la fenêtre.
- Des constantes d'erreur pour le retour en mode « debug ».
- Une structure SCORE qui associe un entier joueur (énumération) à un entier score.
- Une structure CASE qui contiendra un entier "ligne" et un entier "colonne".
- Une structure CLIC qui permettra d'avoir les coordonnées en X et Y du clic, le joueur courant et le menu local.

B) Description en C commentée

Plateau de jeu

- typedef int PLATEAU[TAILLE_PLATEAU][TAILLE_PLATEAU]

Différents états des cases du plateau

- typedef enum { croix_bas = 1, rond_bas = 2, croix_gauche = 3, rond_gauche = 4, croix_haut = 5, rond_haut = 6, croix_droit = 7, rond_droit = 8, vide = 9, surbrillance = 10, tampon = 11 } SYMBOLE;

Différents menus du jeu pour l'affichage

- typedef enum { menuPrincipal = 0 ; menuRegles = 1, menuChoixSymboleS = 2, menuPartie = 3, menuVictoire = 4, menuChoixSymboleM = 5 } MENUS ;

Différents états du jeu pour la gestion des retours de clic

- typedef enum { redirectMenuPrincipal = 0, redirectMenuRegles = 1, redirectMenuChoixSymboleS = 2, redirectMenuPartie = 3, redirectMenuVictoire = 4, redirectMenuChoixSymboleM = 5, redirectContinue = 6, redirectRecommencer = 7, redirectSurbrillance = 8} ETATS ;

Structure qui permet de stocker le score d'un joueur

- typedef struct score {
  int joueur;
  int score ;
  } SCORE;

Structure qui permet de passer facilement d'une fonction à l'autre les coordonnées d'une case du

plateau

- typedef struct case {
  int colonne;
  int ligne;
  } CASE ;

Structure qui permet de passer facilement d'une fonction à l'autre les coordonnées d'un clic sur l'interface graphique

- typedef struct clic {
  int coordX ;
  int coordY ;
  int joueurCourant ;
  int menu ;
  }CLIC ;

3. DESCRIPTION DES FONCTIONS

A) Description des fonctions

- Fonction de décalage d'un symbole

```
int decalage(CASE caseJouee, CASE casePiochee)
{

//Décalage à appliquer
int différenceColonne , différenceLigne ;

différenceColonne = casePiochee.colonne – caseJouee.colonne ;
différenceLigne =  casePiochee.ligne -  casePiochee.ligne ;

//Si le joueur reste sur la même ligne
Si(différenceLigne ==  0 )
{
        Si (différenceColonne <0)
        {
                -> On décale les cases du plateau de jeu de la droite vers la gauche. Cette action est
                répétée en fonction de la valeur absolue de différenceColonne.
                return(valeurOk)
        }
        Si (différenceColonne >0)
        {
                -> On décale les cases du plateau de jeu de la gauche vers la droite. Cette action est
                répétée en fonction de la valeur absolue de différenceColonne.
                return(valeurOk)
```

```
        }
}
//Si le joueur reste sur la même colonne
Si (différenceColonne==  0)
{
        Si ( différenceLigne<0 )
        {
                -> On décale les cases du plateau de jeu du bas vers le haut. Cette action est répétée
                en fonction de la valeur absolue de différenceLigne.
                return(valeurOk)
        }
        Si (différenceLigne >0 )
        {
                -> On décale les cases du plateau de jeu du haut vers le bas. Cette action est répétée
                en fonction de la valeur absolue de différenceLigne.
                return(valeurOk)
        }
}
return(valeurPasOk)
}
```

- Fonction qui détermine les cases possibles de dépôt après une pioche

```
int calculeSurbrillance(CASE casePiochee)
{
        int différenceColonne , différenceLigne ;
        int indiceLigne = casePiochee.ligne ;
        int indiceColonne = casePiochee.colonne;

        On répète cela pour haut, bas, gauche, droite :
                {
                → En partant de la case piochée tant que l'on a pas atteint un des bords du tableau on
                continue de se déplacer pour trouver un bord.
                → Dès que l'on a trouvé un bord on regarde si
                        → La différence (|différenceColonne| ou | différenceLigne| ) est superieur à 1
                        pour éviter qu'il repose son symbole au même endroit
                        → Si elle est bien supérieure à 1 on modifie la valeur dans le plateau pour
                        qu'elle devienne en surbrillance et qu'elle devienne cliquable.
                return(valeurOk).
                }
return(valeurPasOk)
}
```

- Description paramètre spécial :

int [4] coordonneesPlateau ⇒ Coordonnées coin haut gauche et coin bas droite de la zone graphique où s'affiche la grille de jeu.

- Autres fonctions du programme :

- Fonction qui initialise les cases du tableau à 0 :
  int initPlateau()
  Retourne 0 (OK), -1 (KO)

- Fonction qui redirige vers la bonne fonction d'affichage, la détermination vient du paramètre « menu », « coordonnéesPlateau » est la zone graphique où est placée la grille:
  int gestionAffichage(int menu, int [4] coordonneesPlateau)
  Retourne 0 (OK), -1 (KO)

- Fonction qui affiche le menu où le joueur choisit son mode de jeu :
  int afficheMenuPrincipal()
  Retourne 0 (OK), -1 (KO)

- Fonction qui affiche le choix des symboles :
  int afficheMenuSelection()
  Retourne 0 (OK), -1 (KO)

- Fonction qui affiche les règles du jeu :
  int afficheRegles()
  Retourne 0 (OK), -1 (KO)

- Fonction qui affiche la victoire d'un joueur, son score et les différents boutons :
  int afficheVictoire(int joueurGagnant)
  Retourne 0 (OK), -1 (KO)

- Fonction qui affiche le menu de jeu complet :
  int afficheCompletPlateau(int joueurCourant, int [4] coordonneesPlateau)
  Retourne 0 (OK), -1 (KO)

- Fonction qui affiche le plateau (grille, joueur courant, symboles) :
  int affichePlateau(int [4] coordonneesPlateau)
  Retourne 0 (OK), -1 (KO)

- Fonction qui affiche les différentes croix des joueurs :
  int afficheCroixPoint(int pas)
  Retourne 0 (OK), -1 (KO)

- Fonction qui affiche les différents cercles des joueurs :
  int afficheRondPoint(int pas)
  Retourne 0 (OK), -1 (KO)

- Fonction qui dessine une croix :
  int afficheCroix(int pas)
  Retourne 0 (OK), -1 (KO)

- Fonction qui dessine un cercle:
  int afficheRond(int pas)
  Retourne 0 (OK), -1 (KO)

- Fonction qui définit la taille du plateau de jeu en graphique
  int dimensionnePlateau(int [4] coordonneesPlateau)
  Retourne 0 (OK), -1 (KO)

- Fonction qui renvoie le plus petit côté pour obtenir un carré
  int selectionDim(int [4] coordonneesPlateau)
  Retourne un entier XcotePlusPetit

- Fonction qui affiche la grille de jeu :
  int afficheGrille(int pas)
  Retourne 0 (OK), -1 (KO)

- Fonction qui affiche quel joueur doit jouer :
  int afficheJoueurCourant(int joueurCourant)
  Retourne 0 (OK), -1 (KO)

- Fonction qui affiche les scores des joueurs
  int afficheScores()
  Retourne 0 (OK), -1 (KO)

- Fonction qui permet d’afficher les boutons du menu de la partie :
  int afficheBoutons()

Retourne 0 (OK), -1 (KO)

- Fonction qui affiche les cases où le joueur peut reposer son cube en les mettant en surbrillance :
  int afficheSurbrillance(int [4] coordonneesPlateau)
  Retourne 0 (OK), -1 (KO)

- Fonction qui permet de calculer un tour de jeu :
  int calculeTour(int joueurCourant, struct CASE caseJouee, struct CASE casePiochee)
  Retourne 0 (OK), -1 (KO)

- Fonction qui permet de déplacer les cubes dans la grille lorsque le joueur remplit une rangée incomplète :
  int decalage(struct CASE caseJouee, struct CASE casePiochee)
  Retourne 0 (OK), -1 (KO)

- Fonction qui teste s'il y a un gagnant en utilisant les trois fonctions suivantes :
  int testeVictoire(int joueurCourant)
  Retourne 0 (Joueur Gagnant), -1 (jeu continue), -2 (le joueur a perdu au profit de l'adversaire)

- Fonction qui teste si cinq mêmes symboles sont alignés verticalement :
  int testeVictoireLigne(int joueurCourant)
  Retourne 0 (Joueur Gagnant), -1 (jeu continue), -2 (le joueur a perdu au profit de l'adversaire)

- Fonction qui test si cinq mêmes symboles sont alignés diagonalement :
  int testeVictoireDiagonale(int joueurCourant)
  Retourne 0 (Joueur Gagnant), -1 (jeu continue), -2 (le joueur a perdu au profit de l'adversaire)

- Fonction qui test si cinq mêmes symboles sont alignés horizontalement :
  int testeVictoireColonne(int joueurCourant)
  Retourne 0 (Joueur Gagnant), -1 (jeu continue), -2 (le joueur a perdu au profit de l'adversaire)

- Fonction qui permet de déterminer où le joueur peut reposer son cube (modifie une case du tampon dans le plateau 6*6) :
  int calculeSurbrillance(struct CASE casePiochee)
  Retourne 0 (OK), -1 (KO)

- Fonction qui permet de gérer les coordonnées du clic de la souris (renvoie vers la fonction qui

gère ce menu là) :
int recupereClicAffichage(struct CLIC clicJoueur, int [4] coordonneesPlateau)
Retourne un entier qui correspond à l'action à effectuer dans le main (énumération ETATS)

- Fonction qui gère le clic dans les menus (boutons) :
  int clicMenu(struct CLIC clicJoueur)
  Retourne un entier qui correspond à l'action à effectuer dans le main

- Fonction qui gère le clic lors d'une partie (menu de la partie, plateau et boutons) :
  int clicPlateau(struct * CASE caseJouee, struct CLIC clicJoueur, int[4] coordonneesPlateau)
  Retourne un entier qui correspond à l'action à effectuer dans le main, sinon 0 pour bon fonctionnement et remplissage de « caseJouee » pour envoi vers calculeTour.

- Fonction qui teste le symbole (neutre, croix, rond) d'une case en fonction du joueur courant
  int verifieSymboleCase(struct CLIC clicJoueur)
  Retourne 0 (Case autorisée), -1 (Case déjà jouée)

- Fonction qui empêche les joueurs de cliquer sur une des neufs cases au centre de la grille :
  int clicCentral(struct CLIC clicJoueur , int[4] coordonneesPlateau)
  Retourne 0 (Case autorisée), -1 (Case centrale interdite)

- Fonction qui permet de gérer le déplacement des pions (si un joueur est en train de jouer un pion il ne peut pas cliquer sur les boutons du menu) :
  int clicModeDeplacement()
  Retourne 0 (Mode activé), -1 (Mode désactivé)

- Fonction qui permet le redimensionnement de la fenêtre :
  int redimensionnement(int tailleXfenetre, int tailleYfenetre)
  Retourne 0 (OK), -1 (KO)

- fonction qui permet de bloquer le redimensionnement à une taille minimum :
  int redimensionnementMin()
  Retourne 0 (OK), -1 (KO)

- Fonction qui permet d'afficher les coups interdit au joueur courant :
  int afficheDeplacementInterdit()
  Retourne 0 (OK), -1 (KO)

- Fonction qui permet de récupérer une valeur dans le plateau :
  int getCase(PLATEAU plateau, int ligne, int colonne)
  Retourne valeurCase(OK) , -1 (Problème dans la fonction)

- Fonction qui permet de remplir une valeur dans le plateau :
  int setCase(PLATEAU plateau, int ligne, int colonne, int valeur)
  Retourne 0 (OK), -1 (KO)

- Fonction qui permet de modifier un score
  int setScore(SCORE * score, int valeur)
- Retourne 0 (OK), -1 (KO)

- Fonction qui permet de lire le score d'un joueur
  int getScore(SCORE * score)
  Retourne le score du joueur

- Fonction qui renvoie une case générée par l'IA
  struct CASE retourneCase()
  Retourne une CASE

- Fonction qui détermine le coup de l'IA
  int MinMax()
  Retourne un entier correspondant au nœud choisi

- Fonction qui évalue le plateau généré par l'IA
  int evaluePlateau(PLATEAU plateau)
  Retourne un entier de la valeur du coup joué (par rapport au score)

B) Erreurs et oublis