CAHIER DE CONCEPTION GÉNÉRALE

Projet d'algorithmique 2016-2017

Version: CCG_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 générale	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 / PPA / MBO	Ajout fonctions et descriptions

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
SQL	:	Structured Query Language

1. INTRODUCTION

Ce document présente l'architecture de l'algorithme du jeu Quixo dans le cadre du projet de développement en C.
Le programme a pour objectif d'effectuer une simulation informatique du jeu Quixo en se basant sur les règles officielles du jeu.

Ce document de conception général décrit la structure globale du code informatique qui permettra le bon fonctionnement du programme.

2. MODULES FONCTIONNELS

A) Architecture des modules

Les différents modules du programme sont :

- Un module correspondant au moteur du jeu. Fonctions principales de fonctionnement de l'algorithme du jeu en suivant les règles officielles.
- Un module outils qui permettra d'accéder facilement aux structures de données.
- Un module d'affichage qui permettra d'afficher et de jouer au jeu dans une fenêtre graphique.
- Un main qui permettra de lancer le jeu et les appels aux fonctions.

B) Données utilisées par chaque module

Le module d'affichage utilisera toutes les informations sur les joueurs, le plateau courant, les scores, les possibilités de jeu du joueur et les retours de victoire.

Le module moteur s'occupera de gérer les coups, modifier le plateau en fonction de ces coups, l'intelligence artificielle, vérifier la victoire de joueurs et donc, il utilisera le plateau de jeu, les retours de l'affichage sur le clic de boutons.

Le module outils devra gérer les accès aux structures de données.

Le main utilisera les données des autres modules et permettra un échange de ces données entre chaque module.

C) Échange de données entre modules

Le module main récupérera les données du plateau venant du module moteur et lancera les fonctions d'affichage en fonction des menus demandés. Le main lancera aussi les fonctions d'affichage du module d'affichage.

Le module affichage recevra du main les données du plateau et les affichera dans la fenêtre graphique. Il recevra aussi les informations telles que le joueur courant ou les scores.

Enfin, le module moteur recevra les différents coups des joueurs depuis le main et modifiera le plateau de jeu en fonction afin de l'afficher dans le module affichage.

3. STRUCTURES DE DONNÉES

A) Définition des structures de données

Les structures de données qui seront à utiliser pour le Quixo sont :

- Une constante NB_VICTOIRE(5) , TAILLE_PLATEAU(6), ERREURS
- Un tableau 2D d'entiers de TAILLE_PLATEAU * TAILLE_PLATEAU (6*6) cases, plateau de jeu.
- Une énumération comportant les symboles disponibles (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) et tampon(11))
- Constantes pour la taille de la fenêtre
- Une structure CASE qui contiendra un entier "abscisse" et un entier "ordonnée".
- Une structure SCORE qui associe un entier joueur (énumération) à un entier score.
- Une structure CLIC qui permettra d'avoir les coordonnées en X et Y du clic, le joueur courant et le menu local.

B) Action portant sur ces structures de données

Les actions dans la partie moteur seront :

- la vérification des conditions de victoire sur le plateau.
- les fonctions de décalage lors des déplacements sur le plateau.
- les fonctions des coups interdits.
- les fonctions d'initialisation du plateau.

Les actions portant sur la partie graphique :

- les fonctions d'affichages du tableau.
- les fonctions de surbrillance des coups possibles.
- Les fonctions de redimensionnement de la fenêtre.

Les actions portant sur la partie du main seront :

- la gestion des événements à la souris.
- l'appel des fonctions du moteur.
- l'appel des fonctions de la partie graphique.

C) Visibilité des structures de données

- Le tableau 2D (plateau) sera une variable globale.
- L'énumération sera visible dans la partie moteur et dans l'affichage.
- Les constantes seront visibles sur l'ensemble des modules.
- Les SCORES seront définis en variable globale.

4. ARBRE DES FONCTIONS ET FLUX DES DONNÉES

A) Arbre d'appel et flux de données

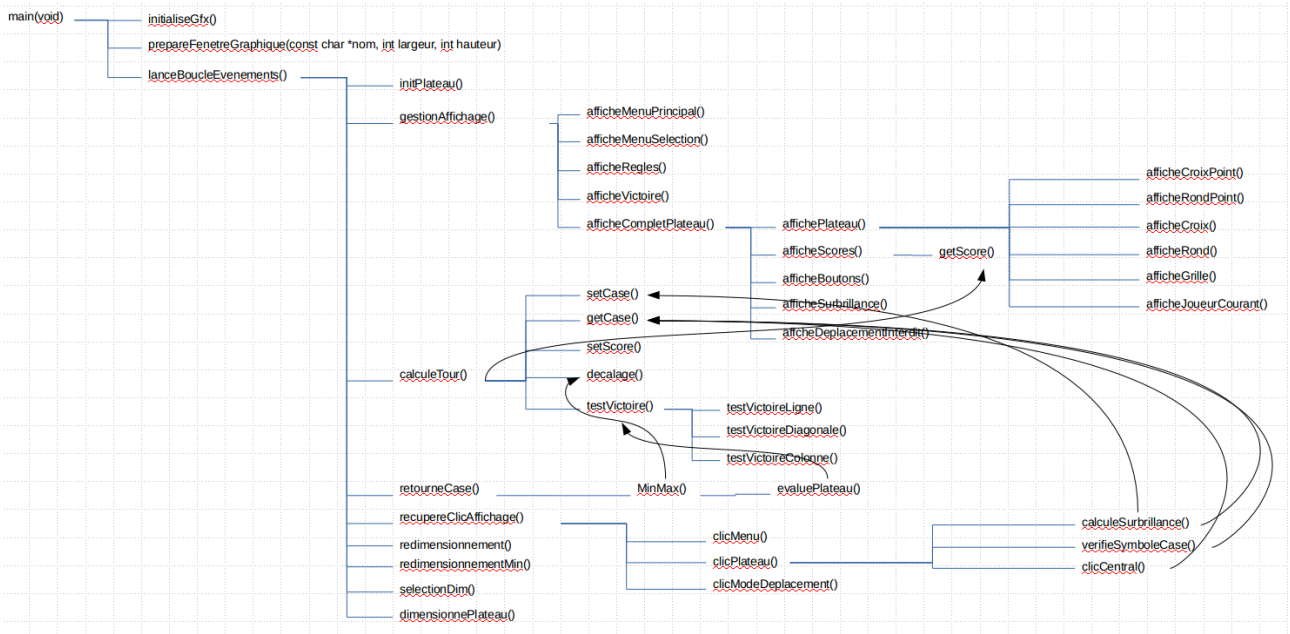

Illustration 1: Arbre d'appel des fonctions

B) Description des fonctions

La liste des fonctions ainsi que la description de chacune d'entre elles est la suivante :

- Fonction qui initialise les cases du tableau à 0 :
  int initPlateau()

- Fonction qui redirige vers la bonne fonction d'affichage :
  int gestionAffichage(int menu, int [4] coordonneesPlateau)

- Fonction qui affiche le menu où le joueur choisit son mode de jeu :
  int afficheMenuPrincipal()

- Fonction qui affiche le choix des symboles :
  int afficheMenuSelection()

- Fonction qui affiche les règles du jeu :
  int afficheRegles()

- Fonction qui affiche la victoire d'un joueur, son score et les différents boutons :

int afficheVictoire(int joueurGagnant)

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

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

- Fonction qui affiche les différentes croix des joueurs :
  int afficheCroixPoint(int pas)

- Fonction qui affiche les différents cercles des joueurs :
  int afficheRondPoint(int pas)

- Fonction qui dessine une croix :
  int afficheCroix(int pas)

- Fonction qui dessine un cercle:
  int afficheRond(int pas)

- Fonction qui définit la taille du plateau de jeu en graphique
  int dimensionnePlateau(int [4] coordonneesPlateau)

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

- Fonction qui affiche la grille de jeu :
  int afficheGrille(int pas)

- Fonction qui affiche quel joueur doit jouer :
  int afficheJoueurCourant(int joueurCourant)

- Fonction qui affiche les scores des joueurs
  int afficheScores()

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

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

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

- 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)

- Fonction qui teste s'il y a un gagnant en utilisant les trois fonctions suivantes :
  int testeVictoire(int joueurCourant)

- Fonction qui teste si cinq mêmes symboles sont alignés verticalement :
  int testeVictoireLigne(int joueurCourant)

- Fonction qui test si cinq mêmes symboles sont alignés diagonalement :
  int testeVictoireDiagonale(int joueurCourant)

- Fonction qui test si cinq mêmes symboles sont alignés horizontalement :
  int testeVictoireColonne(int joueurCourant)

- 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)

- 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)

- Fonction qui gère le clic dans les menus (boutons) :
  int clicMenu(struct CLIC clicJoueur)

- 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)

- Fonction qui teste le symbole (neutre, croix, rond) d'une case en fonction du joueurCourant

int verifieSymboleCase(struct CLIC clicJoueur)

- 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)

- 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()

- Fonction qui permet le redimensionnement de la fenêtre :
  int redimensionnement(int tailleXfenetre, int tailleYfenetre)

- fonction qui permet de bloquer le redimensionnement à une taille minimum :
  int redimensionnementMin()

- Fonction qui permet d'afficher les coups interdit au joueur courant :
  int afficheDeplacementInterdit()

- Fonction qui permet de récupérer une valeur dans le plateau :
  int getCase(PLATEAU plateau, int ligne, int colonne)

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

- Fonction qui permet de modifier un score
  int setScore(SCORE * score, int valeur)

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

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

- Fonction qui détermine le coup de l'IA
  int MinMax()

- Fonction qui évalue le plateau généré par l'IA
  int evaluePlateau(PLATEAU plateau)