**PACMAN PROJECT**



Le but de notre projet est de réaliser le célèbre jeu PacMan en C. Le but du PacMan est de manger tous les pixels disposé dans un labyrinthe fixe. En même temps des fantômes parcourent le labyrinthe pour essayer de tuer PacMan. Lorsque PacMan est touché par un fantôme, PacMan perd une vie. Cependant des pixels bonus (représentés par des carrés plus gros) sont disposés sur toute la carte permettant à PacMan de manger les fantômes. On pourra définir plusieurs niveaux qui joueront sur le nombre de vies que PacMan aura lors du commencement de la partie ou le nombre de fantômes.

Le jeu s’affichera dans la console. Il sera possible d’enregistrer son nom afin de pouvoir établir une liste des meilleurs scores. Le déplacement du personnage s’effectuera à l’aide des touches Z,Q,S et D. Un nombre défini de fantômes sortira d’un endroit à intervalle régulier.

Ce projet vous sera présenté par Paul TREHIOU et Victor SENE actuellement en TC02.

ATTENTION ! Il faut impérativement lire le fichier README.md avant d’utiliser notre programme.

Sommaire

Table des matières

[1 Le Main 3](#_Toc389820018)

[1.1 SetWindow 3](#_Toc389820019)

[1.2 Menu 3](#_Toc389820020)

[1.3 Initialisation 3](#_Toc389820021)

[1.4 Rendu arène 3](#_Toc389820022)

[1.5 Affichage 3](#_Toc389820023)

[1.6 Déplacement 4](#_Toc389820024)

[1.7 lectureScore 4](#_Toc389820025)

[1.8 triScore 4](#_Toc389820026)

[2 Les déplacements 5](#_Toc389820027)

[2.1 Le fonctionnement 5](#_Toc389820028)

[3 Les scores 11](#_Toc389820029)

[4 inscriptionScore 11](#_Toc389820030)

[5 lectureScore 12](#_Toc389820031)

[6 TriScore 12](#_Toc389820032)

[7 affichageBestScore 13](#_Toc389820033)

[8 IA 13](#_Toc389820034)

# Le Main

Le programme appel en premier main.c qui déclare et initialise les éléments principaux du jeu à savoir le terrain et les coordonnées du PacMan et des fantômes. Le PacMan sera représenté par un C et les fantômes par des M. On initialise aussi les tableaux qui recueilleront les scores et pseudo.

Nous avons choisi d’utiliser une structure pour les coordonnés des entités du jeu, facilitant le codage et la compréhension du programme.

Nous utiliserons dans ce programme une nouvelle bibliothèque nous permettant d’utiliser des couleurs pour rendre le visuel plus attrayant.

Notre but fût de coller le plus possible au jeu d’arcade mythique qu’est le PacMan, ainsi s’explique le choix des pseudos de 3 caractères notamment.

## SetWindow

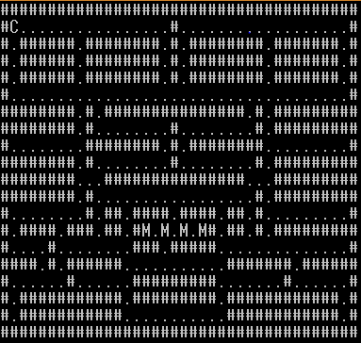
Cette fonction redimensionne la fenêtre

## Menu

Dans ce fichier nous afficherons un menu pour permettre à l’utilisateur de choisir l’action qu’il veut effectuer. Il pourra ainsi se renseigner sur les

## Initialisation

On génère tous les murs, les fantômes, PacMan et les bonus



## Rendu arène

Cette fonction remplace les bords du terrain symbolisés par des #, par des caractères ASCII rendant le parcours plus lisible. L’affectation des symboles est faite dans symboles.h

## Affichage

Le terrain est affiché une première fois

## Déplacement

On lance le jeu

## lectureScore

On lit tous les scores présents dans le fichier score.txt pui on récupère le nombre d’entrées

## triScore

On trie les scores et on les affiches

PROGRAMME PacMan

Variables tableau de caractère terrain [20][38]

S\_coordonees PacMan

S\_coordonees fantomeA

S\_coordonees fantomeB

S\_coordonees fantomeC

S\_coordonees fantomeD

entier mode, maxi

Tableau de 20 chaines de 38 caractères terrain

Tableau de 100 chaines de 4 caractères pseudo[100][4] //Les deux tableaux suivants servent à l'affichage des meilleurs scores

DEBUT

SetWindow(39,25) //forcer une taille de la fenêtre dans le CMD

initialisation du générateur de nombre aléatoire

FAIRE

menu(&mode)

CHOISIR PARMI(mode)

{

case 1 :

initialisation(terrain, &PacMan, &fantomeA, &fantomeB, &fantomeC, &fantomeD) //On génère l'arène

renduarene(terrain) //On effectue un rendu sur l'arène précédement générée afin d'avoir un résultat plus joli

affichage(terrain, 0, 0, 0) //On affiche une première fois le terrain

deplacements(terrain, &PacMan, &fantomeA, &fantomeB, &fantomeC, &fantomeD) //On lance la gestion des déplacements

break

case 2:

system("CLS")

maxi <-- lectureScore(pseuFAIRE, score)

triScore(pseudo, score, maxi)

break

default:

break

}

TANT QUE(mode>0 ET mode<3)

FIN

# Les déplacements

Les déplacements seront déclinés en 2 fichiers : l’un pour les déplacements généraux et l’autre pour les déplacements des fantômes qui doivent avancer sans manger les points du terrain, à l’inverse du Pacman.

Seul l’algorithme pour PacMan est écrit car ils sont très similaires : la fonction de déplacement des fantômes est basée sur celle du PacMan

## Le fonctionnement

On récupère d’abord l’entrée de l’utilisateur à l’aide d’une API Windows

En fonction de la touche pressée on déplace le PacMan. Z pour en haut, Q pour à gauche, S pour le bas et D pour la droite.

On vérifie s’il se dirige vers un point, un mur, un fantôme ou un bonus.

Si le bonus est activé, on permet au PacMan de manger les fantômes, sinon le jeu est fini.

Puis on fait se déplacer les fantômes

Finalement, on affiche la nouvelle frame

Le bonus est ensuite contrôlé : si il a été actif plus de 50 frames, il est désactivé.

Si on perd, on sort de la boucle et l’écran de game over s’affiche

entier deplacements (tableau de 20 chaines de 38 caractères terrain, pointeur vers coordone coordPacMan, pointeur vers Coordone fantomeA, pointeur vers Coordone fantomeB, pointeur vers Coordone fantomeC, pointeur vers Coordone fantomeD)

VARIABLES charactère entree, pointA <-- '.', pointB <-- '.', pointC <-- '.', pointD <-- '.'

entiers score <-- 0, bonus <-- 0, frame <-- 0, c

entiers directionA <-- 0, directionB <-- 0, directionC <-- 0, directionD <-- 0

DEBUT

//Le PacMan se déplace à l'aide des touches z,q,s et d. Le jeu peut être quitté avec Esc

Faire

Lire(entree) //En réalité le getch est remplacé par une API Windows afin de ne pas stopper le jeu à chaque frame

CHOISIR entree parmi

's':

SI (terrain[coordPacMan.i + 1][coordPacMan.j]='.')

ALORS

terrain[coordPacMan.i][coordPacMan.j] <-- ' '

coordPacMan.i <-- coordPacMan.i + 1

terrain[coordPacMan.i][coordPacMan.j] <-- 'C'

score <-- score + 1

SINON SI (terrain[coordPacMan.i + 1][coordPacMan.j]=' ')

ALORS

terrain[coordPacMan.i][coordPacMan.j] <-- ' '

coordPacMan.i <-- coordPacMan.i + 1

terrain[coordPacMan.i][coordPacMan.j] <-- 'C'

SINON SI (terrain[coordPacMan.i + 1][coordPacMan.j]='M' ET non(bonus))

ALORS entree <-- 'Q'

SINON SI(terrain[PacMan.i + 1][PacMan.j] = 'M' ET bonus) //Si c'est un fantôme et que le bonus est activé, PacMan mange le fantôme et gagne 10 points

ALORS

terrain[PacMan.i][PacMan.j] <-- ' ' //La réaparition de PacMan est gérée plus loin car il y a plusieurs fantômes et qu'il faut les gérer au cas par cas

PacMan.i <-- PacMan.i + 1

score <-- score + 10

SINON SI(terrain[PacMan.i + 1][PacMan.j] = point) //Si la case est un bonus on active le bonus

ALORS

terrain[PacMan.i][PacMan.j] <-- ' '

PacMan.i <-- PacMan.i + 1

terrain[PacMan.i][PacMan.j] <-- 'C'

bonus <-- 1

SINON RIEN

FIN SI

FIN SI

FIN SI

FIN SI

FIN SI

'q':

SI (terrain[coordPacMan.i][coordPacMan.j - 1]='.')

ALORS

terrain[coordPacMan.i][coordPacMan.j] <-- ' '

coordPacMan.j <-- coordPacMan.j - 1

terrain[coordPacMan.i][coordPacMan.j] <-- 'C'

score <-- score + 1

SINON SI (terrain[coordPacMan.i][coordPacMan.j - 1]=' ')

ALORS

terrain[coordPacMan.i][coordPacMan.j] <-- ' '

coordPacMan.j <-- coordPacMan.j - 1

terrain[coordPacMan.i][coordPacMan.j] <-- 'C'

SINON SI (terrain[coordPacMan.i][coordPacMan.j- 1]='M' ET non(bonus))

ALORS entree <-- 'Q'

SINON SI(terrain[PacMan.i][PacMan.j - 1] = 'M' ET bonus) //Si c'est un fantôme et que le bonus est activé, PacMan mange le fantôme et gagne 10 points

ALORS

terrain[PacMan.i][PacMan.j] <-- ' ' //La réaparition de PacMan est gérée plus loin car il y a plusieurs fantômes et qu'il faut les gérer au cas par cas

PacMan.j <-- PacMan.j - 1

score <-- score + 10

SINON SI(terrain[PacMan.i][PacMan.j - 1] = point) //Si la case est un bonus on active le bonus

ALORS

terrain[PacMan.i][PacMan.j] <-- ' '

PacMan.j <-- PacMan.j - 1

terrain[PacMan.i][PacMan.j] <-- 'C'

bonus <-- 1

SINON RIEN

FIN SI

FIN SI

FIN SI

'z':

SI (terrain[coordPacMan.i - 1][coordPacMan.j]='.')

ALORS

terrain[coordPacMan.i][coordPacMan.j] <-- ' '

coordPacMan.i <-- coordPacMan.i - 1

terrain[coordPacMan.i][coordPacMan.j] <-- 'C'

score <-- score + 1

SINON SI (terrain[coordPacMan.i - 1][coordPacMan.j]=' ')

ALORS

terrain[coordPacMan.i][coordPacMan.j] <-- ' '

coordPacMan.i <-- coordPacMan.i - 1

terrain[coordPacMan.i][coordPacMan.j] <-- 'C'

SINON SI (terrain[coordPacMan.i - 1][coordPacMan.j]='M' ET non(bonus))

ALORS entree <-- 'Q'

SINON SI(terrain[PacMan.i - 1][PacMan.j] = 'M' ET bonus) //Si c'est un fantôme et que le bonus est activé, PacMan mange le fantôme et gagne 10 points

ALORS

terrain[PacMan.i][PacMan.j] <-- ' ' //La réaparition de PacMan est gérée plus loin car il y a plusieurs fantômes et qu'il faut les gérer au cas par cas

PacMan.i <-- PacMan.i - 1

score <-- score + 10

SINON SI(terrain[PacMan.i - 1][PacMan.j] = point) //Si la case est un bonus on active le bonus

ALORS

terrain[PacMan.i][PacMan.j] <-- ' '

PacMan.i <-- PacMan.i - 1

terrain[PacMan.i][PacMan.j] <-- 'C'

bonus <-- 1

SINON RIEN

FIN SI

FIN SI

FIN SI

'd':

SI (terrain[coordPacMan.i][coordPacMan.j + 1]='.')

ALORS

terrain[coordPacMan.i][coordPacMan.j] <-- ' '

coordPacMan.j <-- coordPacMan.j + 1

terrain[coordPacMan.i][coordPacMan.j] <-- 'C'

score <-- score + 1

SINON SI (terrain[coordPacMan.i][coordPacMan.j + 1]=' ')

ALORS

terrain[coordPacMan.i][coordPacMan.j] <-- ' '

coordPacMan.j <-- coordPacMan.j + 1

terrain[coordPacMan.i][coordPacMan.j] <-- 'C'

SINON SI (terrain[coordPacMan.i][coordPacMan.j + 1]='M' ET non(bonus))

ALORS entree <-- 'Q'

SINON SI(terrain[PacMan.i][PacMan.j + 1] = 'M' ET bonus) //Si c'est un fantôme et que le bonus est activé, PacMan mange le fantôme et gagne 10 points

ALORS

terrain[PacMan.i][PacMan.j] <-- ' ' //La réaparition de PacMan est gérée plus loin car il y a plusieurs fantômes et qu'il faut les gérer au cas par cas

PacMan.j <-- PacMan.j + 1

score <-- score + 10

SINON SI(terrain[PacMan.i][PacMan.j + 1] = point) //Si la case est un bonus on active le bonus

ALORS

terrain[PacMan.i][PacMan.j] <-- ' '

PacMan.j <-- PacMan.j + 1

terrain[PacMan.i][PacMan.j] <-- 'C'

bonus <-- 1

SINON RIEN

FIN SI

FIN SI

FIN SI

SINON Rien

SI(PacMan.i = fantomeA.i && PacMan.j = fantomeA.j)

ALORS

terrain[fantomeA.i][fantomeA.j] <-- 'C'

fantomeA.i <-- 13

fantomeA.j <-- 15

pointA <-- ' '

terrain[fantomeA.i][fantomeA.j] <-- 'M'

FIN SI

SINON SI(PacMan.i = fantomeB.i && PacMan.j = fantomeB.j)

ALORS

terrain[fantomeB.i][fantomeB.j] <-- 'C'

fantomeB.i <-- 13

fantomeB.j <-- 17

pointB <-- ' '

terrain[fantomeB.i][fantomeB.j] <-- 'M'

FIN SI

SINON SI(PacMan.i = fantomeC.i && PacMan.j = fantomeC.j)

ALORS

terrain[fantomeC.i][fantomeC.j] <-- 'C'

fantomeC.i <-- 13

fantomeC.j <-- 19

pointC <-- ' '

terrain[fantomeC.i][fantomeC.j] <-- 'M'

FIN SI

SINON SI(PacMan.i = fantomeD.i && PacMan.j = fantomeD.j)

ALORS

terrain[fantomeD.i][fantomeD.j] <-- 'C'

fantomeD.i <-- 13

fantomeD.j <-- 21

pointD <-- ' '

terrain[fantomeD.i][fantomeD.j] <-- 'M'

FIN SI

//Deplacement des fantomes

directionA <-- ia(\*PacMan, \*fantomeA, terrain, directionA);

directionB <-- ia(\*PacMan, \*fantomeB, terrain, directionB);

directionC <-- ia(\*PacMan, \*fantomeC, terrain, directionC);

directionD <-- ia(\*PacMan, \*fantomeD, terrain, directionD);

//Deplacement des fantomes

pointA <-- deplacementFantome(directionA, fantomeA, pointA, terrain, &entree, bonus)

pointB <-- deplacementFantome(directionB, fantomeB, pointB, terrain, &entree, bonus)

pointC <-- deplacementFantome(directionC, fantomeC, pointC, terrain, &entree, bonus)

pointD <-- deplacementFantome(directionD, fantomeD, pointD, terrain, &entree, bonus)

//On affiche la nouvelle frame ainsi générée

affichage(terrain, score, bonus, frame);

//compteur de frame

SI(bonus)

ALORS

frame <-- frame + 1

FIN SI

//On réinitialise le bonus après 50 frames

SI(frame>50)

ALORS

bonus<--0

frame<--0

FIN SI

Tant que entree =/= 'Q'

//On vide le buffer clavier car la fonction GetKeyState garde toutes les entrées en mémoire et les ressorts au getch suivant

FAIRE

textcolor(12)

ECRIRE(\n\n\n GAME OVER\n\n\n\n)

textcolor(15)

ECRIRE(Pour sauvegarder et afficher votre score appuyer sur la touche espace)

c <-- getch()

system("CLS)

} while (c =/= ' ') //On valide par espace

fscore(score)

FIN

# Les scores

La fonction fscore (contenu dans score.c) est appelée en fin de partie par le programme pour effectuer les tâches telles que la sauvegarde et l’affichage des meilleurs scores. On aura un donc une fonction pour écrire le score dans un fichier .txt (inscriptionScore.c) puis un algorithme effectuant la lecture des scores du fichier et l’inscription de ceux-ci dans un tableau (lectureScore.c) pour finir par un tri (triScore.c).

TriScore.c permet à la fois de classer les scores des différents joueurs, mais il appelle aussi la fonction affichageBestScore.c qui va afficher jusqu’au 10 meilleurs scores contenu dans le fichier. Nous avons prévu aussi un vidage du fichier pour éviter tous les problèmes de dépassement de capacité du tableau des scores.

entier fscore(entier scorePlayer)

DEBUT

entier score[100]={0}, maxi

tableau de 12 charactères name[12]

tableau de 100 charactères pseudo[100][4]

Ecrire("Votre Pseudo (3 caracteres): ")

FAIRE

Lire (name)

TANT QUE (strlen(name)!=3) //test sur la longueur de la chaine de caractères

Ecrire("Votre score : " scorePlayer)

inscriptionScore(name, scorePlayer)

maxi <-- lectureScore(pseudo, score)

triScore(pseudo, score, maxi)

FIN

# inscriptionScore

Cette fonction sert à créer les fichiers de sauvegarde des scores.

On écrit le nouveau score à la fin du fichier

vide inscriptionScore(chaine de charactère name[],entier score)

DEBUT

FILE\* sauv <-- NULL

sauv <-- fopen("score.txt", "a")

SI (sauv != NULL)

ALORS //execution du code dans le cas où il n'y a pas d'erreur

fprintf(sauv, "%s %d\n", name, score); //ecriture du score dans le fichier

fclose(sauv) //fermeture du fichier

SINON

Ecrire("Erreur lors de l'ouverture du fichier pour ecriture") // en cas d'erreurs

FIN SI

FIN

# lectureScore

Cette fonction lit tous les scores présents dans le fichier et les stocks dans des tableaux.

entier lectureScore(tableau de chaine de charactère pseudo[100][4], tableau de 100 entier score[100])

{

FILE\* sauv <-- NULL

sauv <-- fopen("score.txt", "r")

tableau de charactères lecture[4]

entier i <-- 0

SI (sauv != NULL)

ALORS

//execution du code dans le cas où il n'y a pas d'erreur

fseek(sauv, 0, SEEK\_SET) //on se place au début du fichier (on veux être sûr que c'est le cas)

TANT QUE (on est pas à la fin du fichier sauv)

fgets(lecture, 4, sauv)

strcpy(pseudo[i], lecture) //Il faut utiliser ça pour copier des chaines de caractère

fseek(sauv, 1, SEEK\_CUR) // On passe l'espace

fgets(lecture, 4, sauv) //On lit le score

score[i] <-- atoi(lecture) //On convertit les charactère contenu par le tableau lecture en nombre

i <-- i+1

FIN TANT QUE

fclose(sauv) //fermeture fichier

SINON

ECRIRE("Erreur lors de l'ouverture du fichier pour lecture") //message d'erreur

FIN SI

Retourner (i-1)

FIN

# TriScore

Cette fonction trie les tableaux de scores précédemment lus et les envois à la fonction d’affichage

vide triScore(tableau de chaine de charactère pseudo[100][4], tableau de 100 entier score[100], entier maxi)

DEBUT

entier i, j, new\_score

chaine de 4 charactère new\_pseudo[4]

POUR (i=1 à maxi par pas de 1)

new\_score <-- score[i]

strcpy(new\_pseudo, pseudo[i])

j <-- i

TANT QUE (j > 0 ET score[j-1] > new\_score)

score[j] <-- score[j-1];

strcpy(pseudo[j], pseudo[j-1]); // copie d'un chaine dans une autre

j <-- j-1

FIN TANT QUE

score[j] <-- new\_score

strcpy(pseudo[j], new\_pseudo)

FIN POUR

ECRIRE("Le meilleur score est ",score[maxi-1],"detenu par ",pseudo[maxi-1])

FIN

# affichageBestScore

Cette fonction affiche un tableau des meilleurs scores et des pseudos.

# IA

Cette fonction est la tête des fantômes ☺ Elle permet de déterminer les obstacles qui se présentent aux fantômes et de les contourner. Les fantômes se déplacent évidemment en direction du PacMan.

entier ia(S\_coordonees pacman, S\_coordonees fantome, tableau de 20 chaines de 38 charactères terrain[20][38], entier olDirection)

ALORS

entier direction=0 // on a 1=Bas, 2=Gauche, 3=Haut, 4=Droite

SI (olDirection != 1 ET (terrain[fantome.i-1][fantome.j]==' ' OU terrain[fantome.i-1][fantome.j]=='.')) //libre haut

ALORS

SI (olDirection != 4 ET (terrain[fantome.i][fantome.j-1]==' ' OU terrain[fantome.i][fantome.j-1]=='.')) //libre à haut,gauche

ALORS

SI (terrain[fantome.i+1][fantome.j]==' ' OU terrain[fantome.i+1][fantome.j]=='.') //libre haut,gauche,bas

ALORS

SI (pacman.i-fantome.i<=0) //SI pacman en haut

ALORS

SI (pacman.j-fantome.j<=0) //SI pacman à gauche

ALORS

//le pacman ce trouve en haut à gauche du fantôme

SI (pacman.i-fantome.i==0)//test pour savoir si le pacman est sur la même ligne que le fantôme

ALORS direction <-- 2

SINON SI (pacman.j-fantome.j==0)

direction <-- 3

SINON

direction <-- rand()%2+2

FIN SI

SINON

direction <-- 3

FIN SI

SINON //SI pacman en bas

SI (pacman.j-fantome.j<=0) //SI pacman à gauche

ALORS

//le pacman ce trouve en bas à gauche du fantôme

SI (pacman.i-fantome.i==0)

direction <-- 2

SINON SI (pacman.j-fantome.j==0)

direction <-- 3

SINON

direction <-- rand()%2+1

FIN SI

SINON

direction <-- 1

FIN SI

FIN SI

SINON SI (terrain[fantome.i][fantome.j+1]==' ' OU terrain[fantome.i][fantome.j+1]=='.') //libre à haut,gauche,droite

SI (pacman.i-fantome.i<=0) //SI pacman en haut

ALORS

SI (pacman.j-fantome.j>=0) //SI pacman à droite

ALORS

//le pacman ce trouve en haut à droite du fantôme

SI (pacman.i-fantome.i==0) //SI pacman sur même ligne

ALORS

direction <-- 4

SINON SI (pacman.j-fantome.j==0)//SI pacman sur même colonne

direction <-- 3

SINON

direction <-- rand()%2+3

FIN SI

SINON SI (pacman.j-fantome.j<=0) //SI pacman à gauche

direction <-- 2

SINON

direction <-- 3

FIN SI

FIN SI

SINON

direction <-- rand()%2+2

FIN SI

SINON SI (olDirection != 2 ET (terrain[fantome.i][fantome.j+1]==' ' OU terrain[fantome.i][fantome.j+1]=='.')) //libre à haut, droite

SI (terrain[fantome.i+1][fantome.j]==' ' OU terrain[fantome.i+1][fantome.j]=='.') //libre haut,droite,bas

ALORS

SI (pacman.i-fantome.i<=0) //SI pacman en haut

ALORS

SI (pacman.j-fantome.j>=0) //SI pacman à droite

ALORS

//le pacman ce trouve en haut à droite du fantôme

SI (pacman.i-fantome.i==0) //SI pacman sur même ligne

ALORS

direction <-- 4

SINON SI (pacman.j-fantome.j==0)//SI pacman sur même colonne

direction <-- 1

SINON

direction <-- rand()%2+3

FIN SI

SINON

direction <-- 3

FIN SI

SINON //SI pacman en bas

SI (pacman.j-fantome.j>=0) //SI pacman à droite

ALORS

//le pacman ce trouve en bas à droite du fantôme

SI (pacman.i-fantome.i==0)

ALORS

direction <-- 4

SINON SI (pacman.j-fantome.j==0)

direction <-- 1

SINON

SI (rand()%2==0)

ALORS

direction <-- 1

SINON

direction <-- 4

FIN SI

FIN SI

SINON

direction <-- 1

FIN SI

FIN SI

SINON SI (terrain[fantome.i][fantome.j+1]==' ' OU terrain[fantome.i][fantome.j+1]=='.') //libre à haut,droite, gauche

SI (pacman.i-fantome.i<=0) //SI pacman en haut

ALORS

SI (pacman.j-fantome.j>=0) //SI pacman à droite

ALORS

//le pacman ce trouve en haut à droite du fantôme

SI (pacman.i-fantome.i==0) //SI pacman sur même ligne

ALORS

direction <-- 4

SINON SI (pacman.j-fantome.j==0)//SI pacman sur même colonne

direction <-- 3

SINON

direction <-- rand()%2+3

FIN SI

SINON

direction <-- 2

FIN SI

FIN SI

FIN SI

SINON SI (olDirection != 3 ET (terrain[fantome.i+1][fantome.j]==' ' OU terrain[fantome.i+1][fantome.j]=='.')) //libre en haut, bas

SI (pacman.i-fantome.i<=0)

ALORS

direction <-- 3

SINON

direction <-- 1

FIN SI

SINON

direction <-- 3

FIN SI

//FIN SI LIBRE HAUT

SINON SI (olDirection != 3 ET (terrain[fantome.i+1][fantome.j]==' ' OU terrain[fantome.i+1][fantome.j]=='.')) //libre bas

SI (olDirection != 4 ET (terrain[fantome.i][fantome.j-1]==' ' OU terrain[fantome.i][fantome.j-1]=='.')) //libre bas, gauche

SI (terrain[fantome.i][fantome.j+1]==' ' OU terrain[fantome.i][fantome.j+1]=='.') //libre bas, gauche, droite

ALORS

SI (pacman.i-fantome.i>=0) //SI pacman en bas

ALORS

SI (pacman.j-fantome.j<=0) //SI pacman à gauche

ALORS

//pacman en bas à gauche

SI (pacman.i-fantome.i==0) //SI pacman sur même ligne

ALORS

direction <-- 2

SINON SI (pacman.j-fantome.j==0)//SI pacman sur même colonne

direction <-- 1

SINON

direction <-- rand()%2+1

FIN SI

SINON

//pacman en bas à droite

SI (pacman.i-fantome.i==0) //SI pacman sur même ligne

ALORS

direction <-- 4

SINON SI (pacman.j-fantome.j==0)//SI pacman sur même colonne

direction <-- 1

SINON

SI (rand()%2==0)

ALORS

direction <-- 1

SINON

direction <-- 4

FIN SI

FIN SI

FIN SI

FIN SI

FIN SI

SINON

direction <-- 1

FIN SI

//FIN SI LIBRE BAS

SINON SI (olDirection != 4 ET (terrain[fantome.i][fantome.j-1]==' ' OU terrain[fantome.i][fantome.j-1]=='.')) //Libre gauche

SI (terrain[fantome.i][fantome.j+1]==' ' OU terrain[fantome.i][fantome.j+1]=='.') //Libre gauche, droite

ALORS

SI (pacman.j-fantome.j<=0) //SI pacman à gauche

ALORS

direction <-- 2

SINON //SI Pacman à droite

direction <-- 4

FIN SI

SINON

direction <-- 2

FIN SI

SINON SI (olDirection != 2 ET (terrain[fantome.i][fantome.j+1]==' ' OU terrain[fantome.i][fantome.j+1]=='.')) //Libre droite

ALORS

direction <-- 4

SINON

direction <-- rand()%4+1

FIN SI

Retourner direction