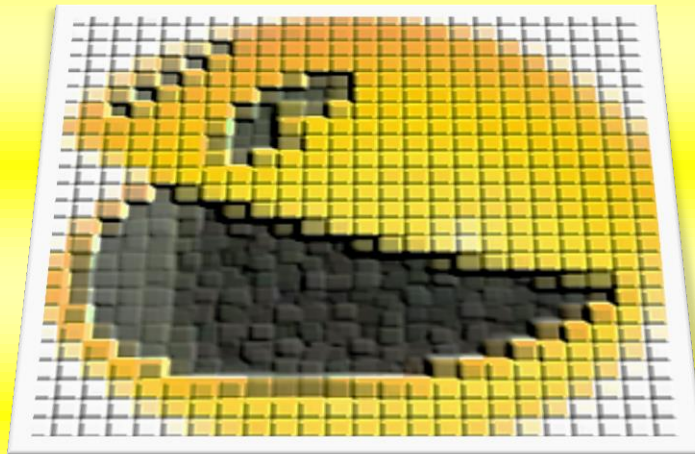


Rapport du TP4 : PACMAN



Maccarinelli Chloé L2-infomatique

Sommaire :

- *Histoire du jeu*
- *Etapes de réalisation*
 - *Gestion du labyrinthe*
 - *Gestion de Pacman*
 - *Programme principale*
 - *Les fantômes :
déplacements aléatoires, leur
mémoire*
 - *Intelligence artificielle*
- *Améliorations*

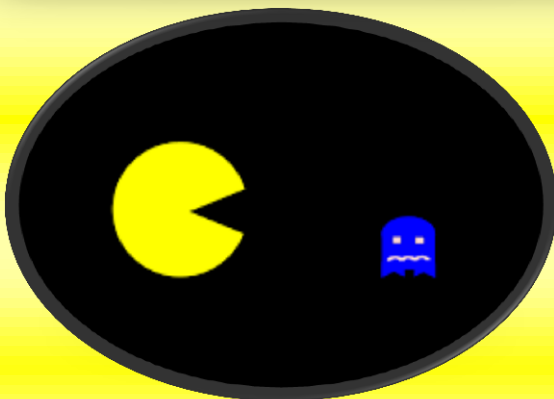
Histoire du jeu



Pacman fut créé en 1980 au Japon par Toyu Iwatani. Les règles sont simples : Pac-Man se trouve dans un labyrinthe et il doit manger toutes les pacgums avant que les fantômes ne l'attrapent. Certaines pacgums rendent Pacman invulnérable et lui permettent de détruire les fantômes. Des bonus sous forme de fruits et d'une clé apparaissent au fur et à mesure de l'avancement du jeu. Pacman doit manger 244 points de la même couleur, tout en évitant les quatre fantômes (Inky, Pinky, Blinky et Clyde).

Pacman est un des jeux de plateforme les plus connus au monde et continu d'évoluer encore aujourd'hui, on connaît évidemment Super-Pacman ou encore Ms Pacman.

C'est entre 1982 et 1984 que Pacman est à son apogée, il va en effet connaître une baisse de popularité en 1984 mais c'est sans compter qu'à partir de 1990 Pacman renaît en apparaissant sur les nouvelles consoles ; Playstation, PSP, Playstation2, PC, Xbox...



Etapas de réalisation

- Gestion du labyrinthe

Cette étape consiste à créer un labyrinthe et à y stocker des informations.

Pour cela on a besoin de définir une structure de données Labyrinthe :

```
typedef struct{
    char** lab;
    int lin;
    int col;
    int nbGomme;
}labyrinthe;
```

1^{ère} méthode :

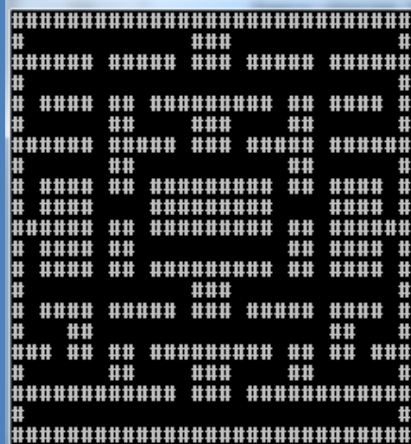
createEmptyLab () :

```
int createEmptyLab( labyrinthe *l, int lin, int col)
{
    int i,j;

    if(lin>0 && col>0)
    {
        (*l).col = col;
        (*l).lin = lin;
        (*l).lab = malloc(lin*sizeof(char*));

        for(i=0;i<lin;i++)
        {
            (*l).lab[i] = malloc(col * sizeof(char));

            for(j=0;j<col;j++)
            {
                (*l).lab[i][j]=' ';
            }
        }
        return 1;
    }
    else
    {
        return 0;
    }
}
```



Ici on crée le labyrinthe en lui donnant en paramètre le nombre de lignes et de colonnes(21,29)

2^{ème} méthode :

freeLab() :

```
void freeLab(labyrinthe *l){ /*libere l'espace allouer */
    int i;
    for(i=0; i < (*l).lin; i++)
    {
        free( (*l).lab[i] );
        free( (*l).lab );
    }
}
```

On obtient ici la même chose mais la procédure prend en paramètre un labyrinthe et non pas le nombre de lignes et de colonnes.

creatFileLab () :

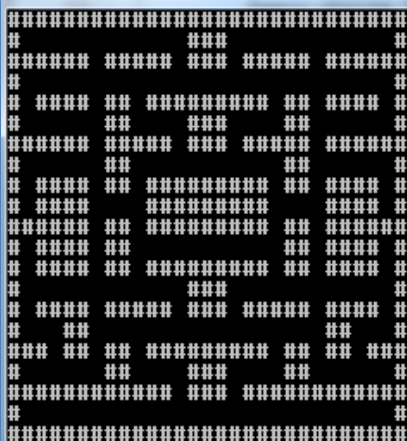
On initialise le
labyrinthe
à l'aide du fichier
texte
Labyrinthe.txt.

```
int createFileLab(char * filename, labyrinthe * l)
{
    int i=0,j,lin,col;

    (*l).nbGomme=0;
    FILE* fichier =NULL;
    char chaine[TAILLE_MAX]="";

    sprintf(chaine,"Labyrinthes/%s.txt",filename);
    fichier = fopen(chaine,"r");
    if(fichier == NULL)
    {
        return -1;
    }

    else
    {
        fscanf(fichier,"%d %d",&lin,&col);
        createEmptyLab(l,lin,col);
        fgets(chaine,TAILLE_MAX,fichier);
    }
}
```



Résultat de la
procédure.

```

void printScreen(labyrinthe * l)
{
    if (system("CLS")) system("clear");
    int i,j;

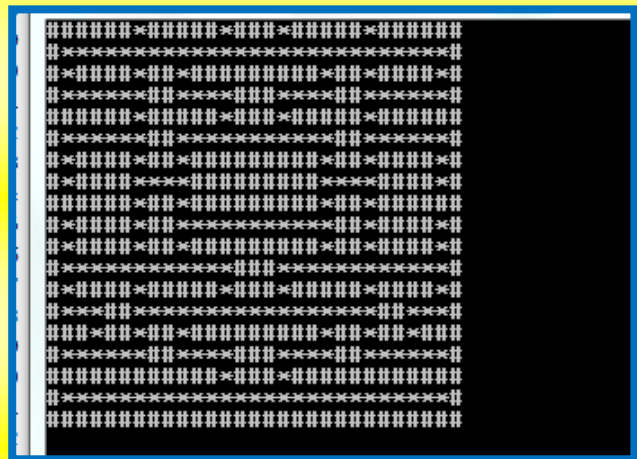
    for(i=0;i<(*l).lin;i++)
    {
        for(j=0;j<(*l).col;j++)
        {
            printf("%c", (*l).lab[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}

```

printScreen () :

On affiche le labyrinthe en haut à gauche et on réinitialise à chaque fois que l'on tape sur une touche.

Enfin le labyrinthe est initialisé avec les pacgums :



- Gestion de Pacman

Ici on veut gérer Pacman, c'est-à-dire gérer ses déplacements (haut, bas, gauche, droite)

La structure de

```
typedef struct{
    int x;
    int y;
    int nbgommes;
    int nbmort;
    int nbvie;
    int super;
    int point;
    bool mange;
    char cible;
    char **pac;
}pacman;
```

données :

Initialisation de
Pacman :

```
void initPacman(pacman *p,int x,int y,labyrinthe *l){
    int i,j;
    if(x>0 && y>0){
        (*p).x = x;
        (*p).y = y;
        (*p).pac = malloc(((*l).lin*sizeof(char*)));
        for(i=0;i<(*l).lin;i++){
            (*p).pac[i] = malloc((*l).col * sizeof(char));
            for(j=0;j<(*l).col;j++){
                (*p).pac[i][j]=' ';
            }
        }
        (*l).lab[x][y]='G';
        (*p).nbvie=3;
        (*p).nbmort=0;
        (*p).nbgommes=0;
        //(*p).super=0;
    }
}
```

Déplacements :

En haut : (O) :

```
bool movePacmanTop(pacman *p,labyrinthe *l){
    int i=(*p).x;
    int j=(*p).y;
    switch ((*l).lab[i-1][j]){
        case '#':{(*l).lab[i][j]='G';break;}
        case '*':{(*p).x--;(*p).nbgommes++;(*l).lab[i][j]=' ';(*l).lab[i-1][j]='G';(*p).point++;break;}
        case ' ':{(*p).x--;(*l).lab[i][j]=' ';(*l).lab[i-1][j]='G';break;}
        case '&':{(*p).nbvie--;initPacman(p,1,1,1);break;}
        case '@':{(*p).nbvie--;initPacman(p,1,1,1);break;}
    }
}
```

En bas : (L) :

```
bool movePacmanBottom(pacman *p, labynthe *l) { /*deplacement de pacman vers le bas */
    int i = (*p).x;
    int j = (*p).y;
    switch ((*l).lab[i+1][j]){
        case '#':{(*l).lab[i][j]='G';break;}
        case '*':{(*p).x++;(*p).nbgommes++;(*l).lab[i][j]=' ';(*l).lab[i+1][j]='G';(*p).point++;break;}
        case ' ':{(*p).x++;(*l).lab[i][j]=' ';(*l).lab[i+1][j]='G';break;}
        case '|':{(*p).x=1;if ((*l).lab[(*p).x][j]!='*') {(*p).nbgommes++;} (*l).lab[i][j]=' ';(*l).lab[(*p).x][j]='G';break;}
        case '&':{(*p).nbvie--;initPacman(p,1,1,1);break;}
        case '@':{(*p).nbvie--;initPacman(p,1,1,1);break;}
    }
}
```

A gauche : (K) :

```
bool movePacmanLeft(pacman *p,labyrinthe *l){          /*deplacement de pacman vers la gauche */
    int i=(*p).x;
    int j=(*p).y;
    switch ((*l).lab[i][j-1]){
        case '#':{(*l).lab[i][j]='G';break; ;}
        case '*':{(*p).y--;(*p).nbgommes++;(*l).lab[i][j]=' ';(*l).lab[i][j-1]='G';(*p).point++;break;}
        case ' ':{(*p).y--;(*l).lab[i][j]=' ';(*l).lab[i][j-1]='G';break;}
        case '-':{(*p).y=(*l).col-2;if ((*l).lab[i][(*p).y]!='*'){(*p).nbgommes++;(*l).lab[i][j]=' ';(*l).lab[i][(*p).y]='G';break;}
        | case '&':{(*p).nbvie--;initPacman(p,l,1,1);break;}
        case '@':{(*p).nbvie--;initPacman(p,l,1,1);break;}
    }
}
```

A droite : (M) :

```
bool movePacmanRight(pacman *p, labyrinthe *l){ /*deplacement de pacman vers la droite */
    int i=(*p).x;
    int j=(*p).y;
    switch ((*l).lab[i][j+1]){
        case '#':{(*l).lab[i][j]='G';break;}
        case '*':{(*p).y++;(*p).nbgommes++;(*l).lab[i][j]=' ';(*l).lab[i][j+1]='G';(*p).point++;break;}
        case ' ':{(*p).y++;(*l).lab[i][j]=' ';(*l).lab[i][j+1]='G';break;}
        case '-':{(*p).y++;if ((*l).lab[i][(*p).y]=='*'){(*p).nbgommes++;}(*l).lab[i][j]=' ';(*l).lab[i][(*p).y]='G';break;}
        case '&':{(*p).nbvie--;initPacman(p,1,1,1);break;}
        case '@':{(*p).nbvie--;initPacman(p,1,1,1);break;}
    }
}
```

Pacman=G

Ici pacman sera toujours initialisé en haut à gauche du labyrinthe.





Quand il passe sur une Pacgum il la mange et la case se vide. Son nombre de Pacgums est compté et est incrémenté à chaque fois qu'il en mange une.

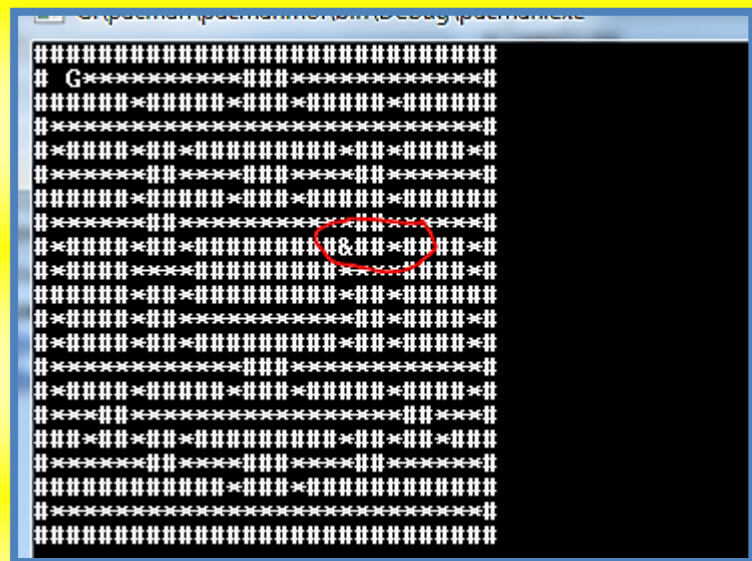
- Les fantômes : déplacements aléatoires, leur mémoire

Structure de données :

```
typedef struct{
    int lin,col;
    char memCase;
    labyrinthe memoire;
}Fantome;
```

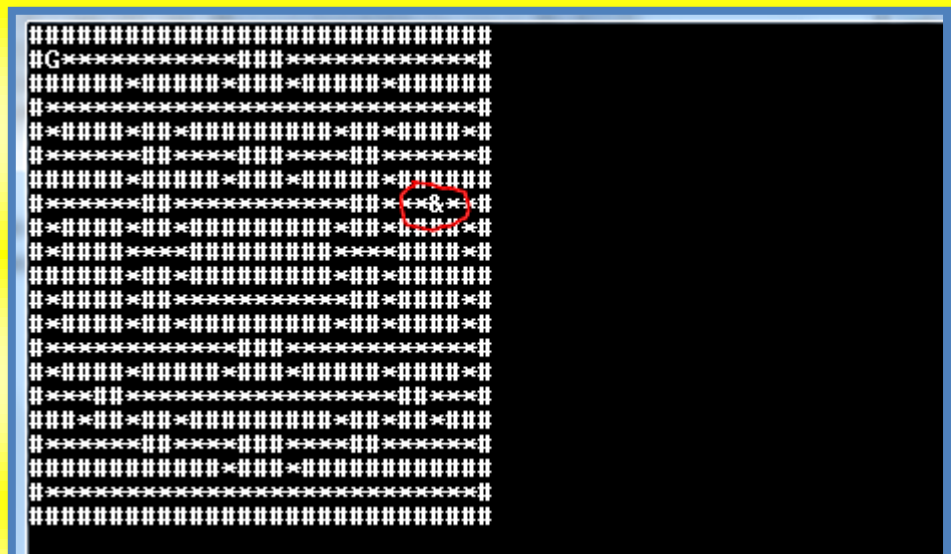
On initialise la position du fantôme au hasard :

initFantomeRand() :



On relance le terminal et on obtient :

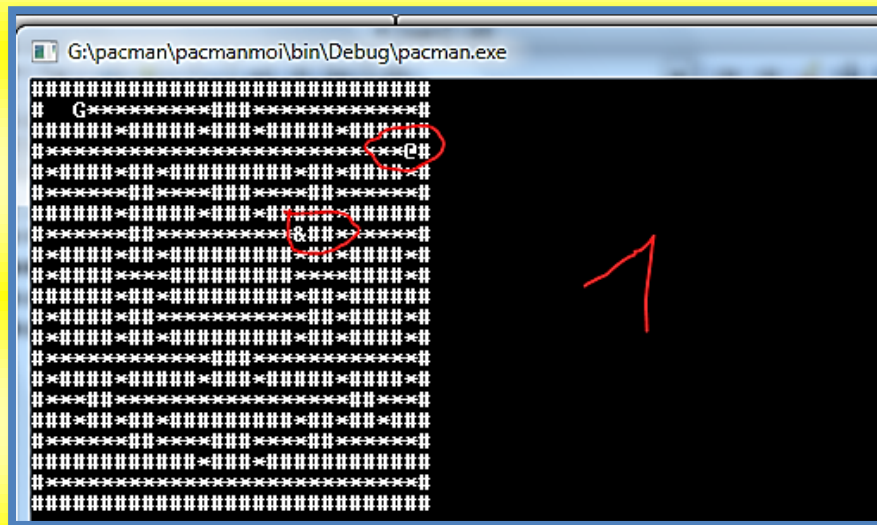
La position est bien différente.



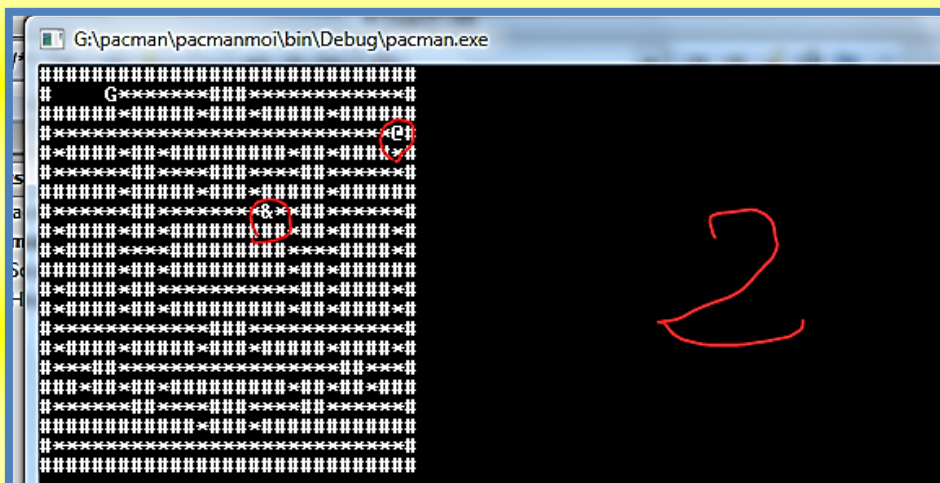
procédure

moveFantomeRand() :

```
void moveFantomeRand(Fantome *f, laby-inthe *l){
    int sens, moveLin, moveCol;
    bool test= false;
    while(!test){
        sens = rand()%4;
        switch(sens){
            case 0: {moveCol = (*f).col+1; moveLin=(*f).lin; break;} /*à droite*/
            case 1: {moveLin = (*f).lin+1; moveCol=(*f).col; break;} /*en bas*/
            case 2: {moveCol = (*f).col-1; moveLin=(*f).lin; break;} /*à gauche*/
            case 3: {moveLin = (*f).lin-1; moveCol=(*f).col; break;} /*en haut*/
        }
        if((*l).lab[moveLin][moveCol] != '#'){
            test = true;
        }
    }
}
```



On déplace le fantôme.



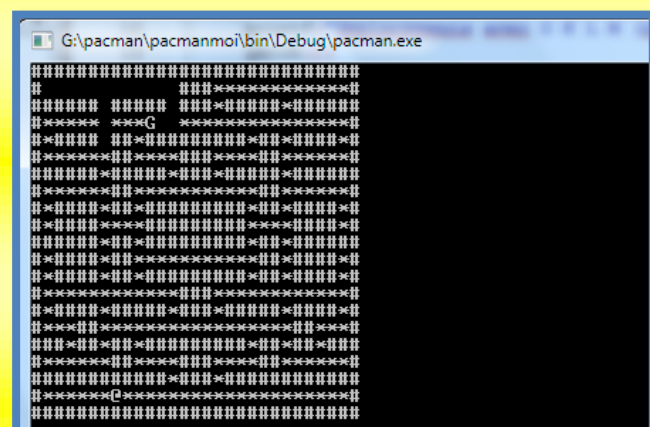
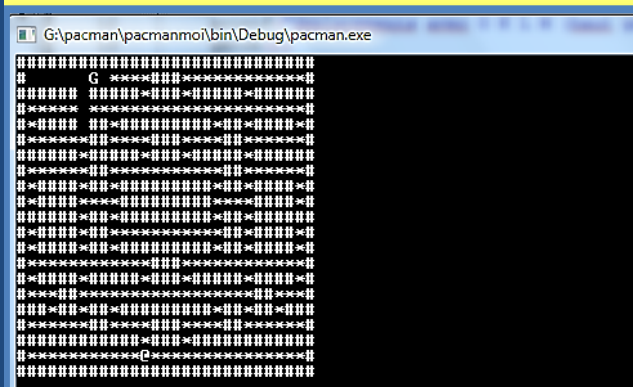
Le fantôme & est celui initialisé dans initFantomeRand() et le fantôme @ est celui initialisé dans initFantomeMem().

Avec les screens précédent on voit bien que les fantômes ont été déplacés après que pacman ait effectué deux déplacements.

On va désormais créer une « mémoire » pour les fantômes afin qu'ils ne stagnent pas dans la même zone du labyrinthe.

copieLab() :

```
void copieLab(labyrinthe l1,labyrinthe *l2)
{
    int i,j;
    createEmptyLab(l2,l1.lin,l1.col);
    for(i=0;i<l1.lin;i++)
    {
        for(j=0;j<l1.col;j++)
        {
            if(l1.lab[i][j]=='#')
            {
                (*l2).lab[i][j]=l1.lab[i][j];
            }
            else
            {
                (*l2).lab[i][j]=' ';
            }
        }
    }
}
```



moveFantomeMem() :

```
void moveFantomeMem(Fantome *f, labyrinthe *l){
    int sens, movelin, movecol;
    srand(time(NULL));
    bool test= false;
    (*l).lab[(*f).lin][(*f).col]=(*f).memCase;
    while(!test){
        sens = rand()%4;
        switch(sens){
            case 0: movecol = (*f).col+1;movelin=(*f).lin;break;
            case 1: movelin = (*f).lin+1;movecol=(*f).col;break;
            case 2: movecol = (*f).col-1;movelin=(*f).lin;break;
            case 3: movelin = (*f).lin-1;movecol=(*f).col;break;
        }
        if((*f).memoire.lab[movelin][movecol] == ' '){
            test = true;
        }
        else if((*f).memoire.lab[movelin][movecol] != '#' && rand()%5==0){
            test = true;
        }
    }

    (*f).memoire.lab[(*f).lin][(*f).col]='9';

    refreshFantomeMem(f, (*l));
    if((*l).lab[movelin][movecol] != '%' && (*l).lab[movelin][movecol] != '@'){
        (*f).memCase=(*l).lab[movelin][movecol];
    }
    (*f).lin=movelin; (*f).col=movecol;
    (*l).lab[movelin][movecol] = '@';
}
```

refreshFantomeMem() :

```
void refreshFantomeMem(Fantome *f, labyrinthe l){
    int i,j;
    for(i=0;i<l.lin;i++){
        for(j=0;j<l.col;j++){
            switch((*f).memoire.lab[i][j]){
                case '0': (*f).memoire.lab[i][j]=' ';break;
                case '#':break;
                case ' ':break;
                default: (*f).memoire.lab[i][j]--;
            }
        }
    }
}
```

- Intelligence artificielle

On veut désormais que les fantômes chassent Pacman. Il doit explorer tous les chemins possibles menant à Pacman en partant de sa propre position. Pour cela on crée une fonction qui renvoie « true » si il existe un chemin vers Pacman « false » sinon.

searchPath() :

```
bool searchPath(labyrinthe *l, int lin, int col, int cont)
{
    if(getCase(l, lin, col) == 'G' || getCase(l, lin, col) == 'C')
    {
        return(true);
    }

    if((getCase(l, lin, col) == '#') || (getCase(l, lin, col) == 'S'))
    {
        return(false);
    }

    if(cont > 10)
    {
        return(false);
    }

    setCase(l, lin, col, 'S');

    if(searchPath(l, lin+1, col, cont+1))
    {
        return(true);
    }

    if(searchPath(l, lin-1, col, cont+1))
    {
        return(true);
    }

    if(searchPath(l, lin, col+1, cont+1))
    {
        return(true);
    }

    if(searchPath(l, lin, col-1, cont+1))
    {
        return(true);
    }

    return(false);
}
```

Cette procédure prend en paramètre le labyrinthe et la position (lin,col) du fantôme. Pour ne pas explorer le même chemin plusieurs fois celui qui mène à Pacman est marqué par « S » et ceux qui ne mènent pas à Pacman seront considérés comme des murs.

Puisque Pacman se déplace, cette opération est effectuée à chaque tour.

Améliorations

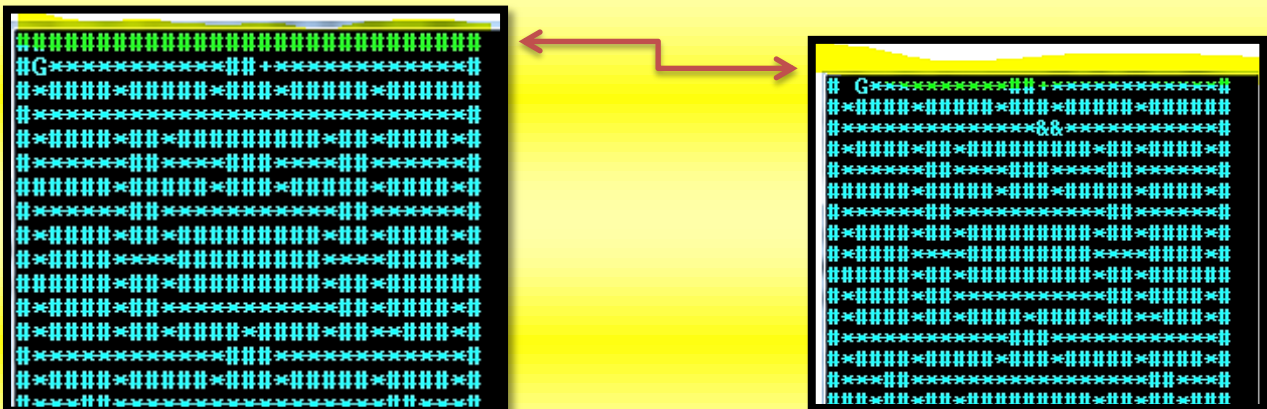
- Pour améliorer le jeu et pour qu'il ressemble au célèbre jeu Pacman, on va permettre à Pacman d'être invulnérable aux fantômes pendant quelques instant ; pour cela il suffit à Pacman de manger une pacgomme « + ».

PROBLEMES :

Ici, pacman devient « super » seulement quand il est sur la superpacgomme; il ne le reste pas

Autre problèmes :

- la gestion de la vie restante ne fonctionne pas.
Pacman se réinitialise alors qu'il devrait être mort, de plus lorsqu'il touche un fantôme il reste sur place bien que un autre pacman se réinitialise en haut à gauche. Ces problèmes viennent de ma procédure rencontre dans pacman.c
- Au début je n'avais pas ce problème mais en modifiant mon code pour la gestion des niveaux la première ligne délimitant le labyrinthe disparaît. Je ne vois pas d'où cela vient.



- Ensuite on permet au joueur de choisir son niveau :

Niveau 1 : Les fantômes se déplacent aléatoirement (moveFantomeRand)

Niveau 2 : les fantômes ont une mémoire (moveFantomeMem)

Niveau 3 : les fantômes recherchent pacman (searchPath), le labyrinthe est différent.

L'interface graphique :

La SDL ne fonctionnant pas j'ai décidé d'utiliser quelques bibliothèques de codeblocks uniquement.

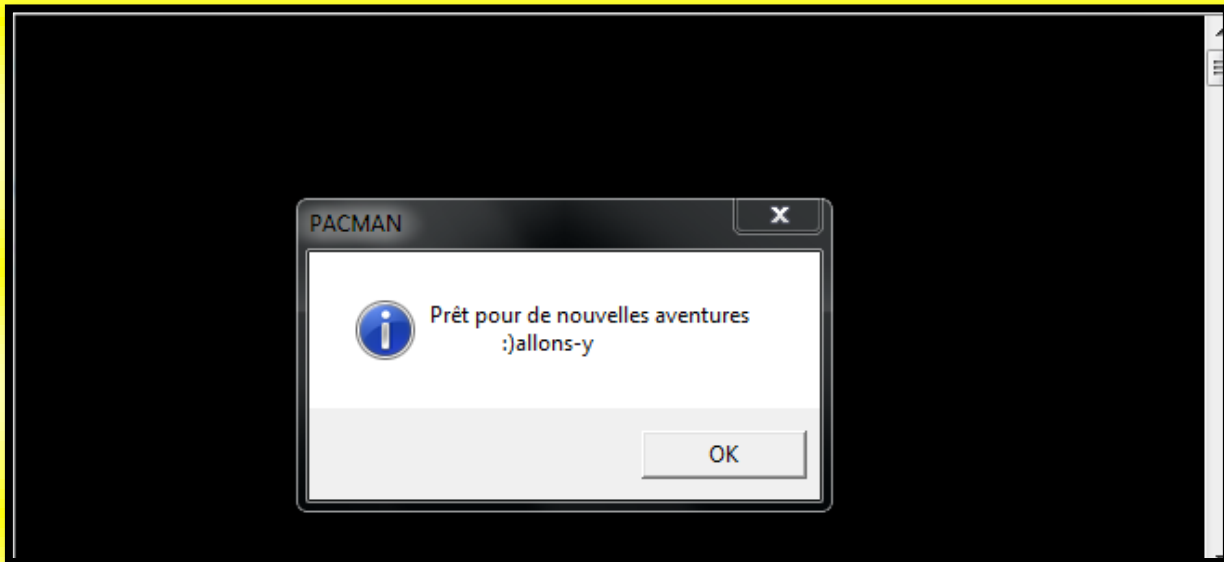
(windows.h, tchar.h)

je modifie les couleurs, notamment dans le printScreen, crée des boîtes de dialogues, modifie le son, création de mot de passe etc.

Les principales améliorations :

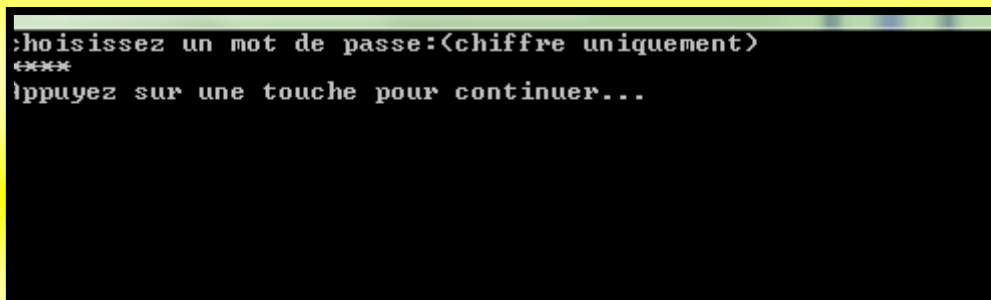
Génération de fenêtres :

- Des fenêtres apparaissent pour des indications appuyer sur entrée pour passer.
- Sons à chaque apparition.



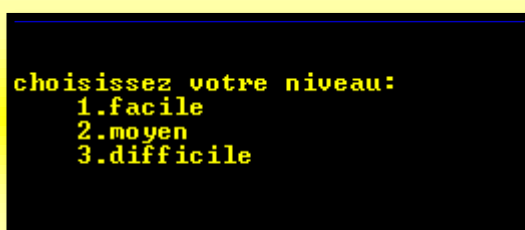
Demande de mot de passe :

- Chiffre uniquement (c'est juste pour personnaliser chaque partie)



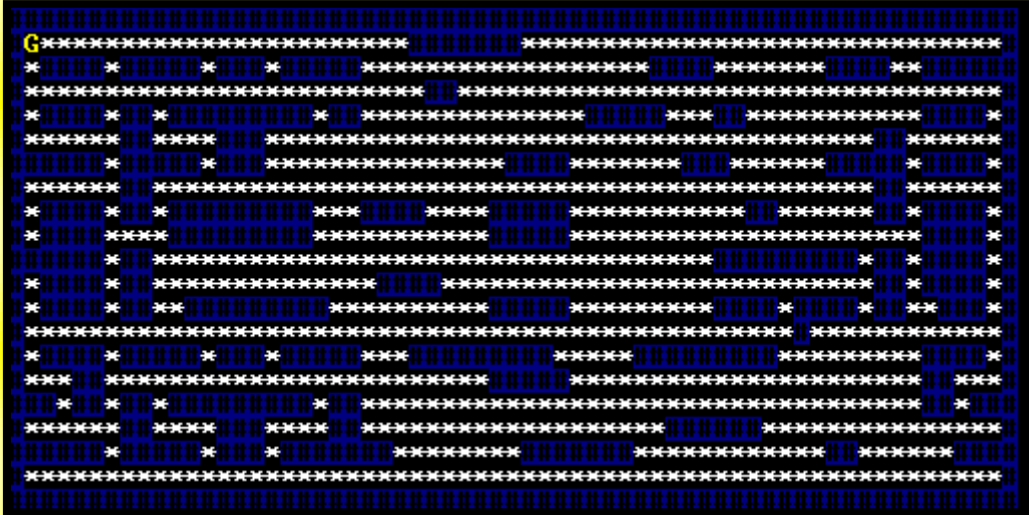
Choix de niveau :

- Chaque niveau (3 au total) à sa spécificité comme développé au-dessus

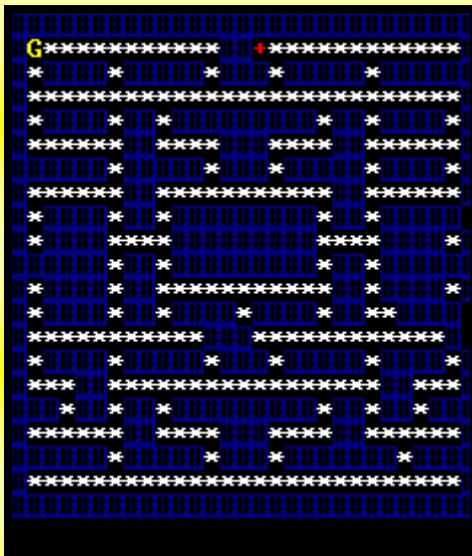


1^{er} labyrinthe : niveau1 et 2

- ➔ Couleurs pour chaque élément
- ➔ Sons à chaque fois que pacman mange une pacgommme



2^{ème} labyrinthe : niveau 3



- ➔ Ajout de la fonction kbhit() pour que pacman bouge tout seul par contre cela fait beuguer le jeu une fois sur 3 il faut relancer le jeu pour que ça ne beug pas mais ça dépend. J'ai donc mis cette fonction uniquement au niveau 2 et 3. *(pour cette fonction c'est un collègue de classe qui m'a conseillé).*