3 Le graphisme (1ère partie)

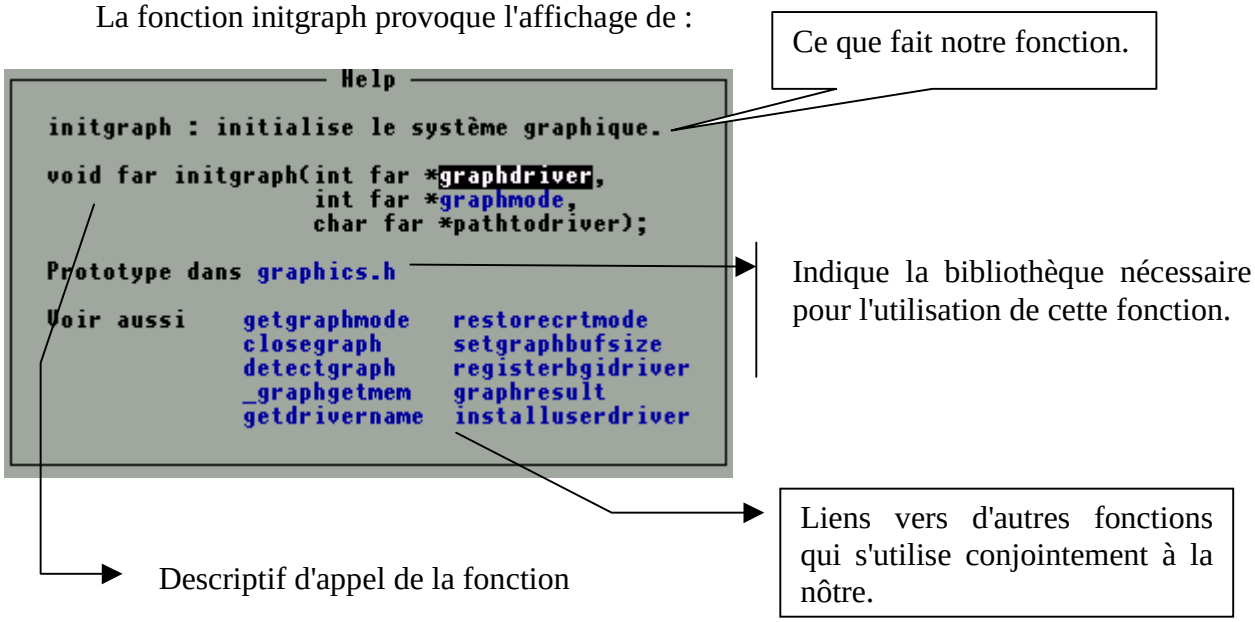
Ras le bol de l'alphanumérique et du 25 * 80 de grand papa… Vive le VGA enfin c'est déjà démodé mais avec notre Turbo C 2.0 c'est tout ce que nous pouvons faire. A noter tout de même que la plupart des jeux d'action Doom Like ont été écrits en MCGA (300*200*256 couleurs !!!).

3.1 Mise en fonction du graphisme & utilisation de l'aide

3.1.1 Utilisation de l'aide

Nous allons apprendre à utiliser l'aide en ligne de Turbo C 2.0.
L'aide en ligne s'affiche lors de l'appui simultané de Ctrl et F1 sur un mot clé.

Exemple :
La fonction initgraph provoque l'affichage de :



Le préfixe far signifie une valeur longue du pointeur (peu importe). Pour notre connaissance générale, il faut simplement savoir que c'est comme si les pointeurs normaux ne pouvaient adresser du courrier que sur la France et les pointeurs far sur le Monde. Les pointeurs far ont disparu avec les compilateurs 32 bits.

Sinon, on retrouve tous les termes précédemment employés …

La bibliothèque nécessaire nous indique qu'il nous faudra ajouter :
#include <graphics.h> à la liste.

Maintenant à nous de jouer.

3.1.2 Les fonctions à utiliser

Voici les fonctions que l'on va utiliser dans un premier temps :

```
initgraph (…)
```

```
graphresult()
closegraph ()
```

Exercice :

En utilisant l'aide en ligne, essayez de comprendre l'utilisation des 3 fonctions.

Explications :

Par défaut, les programmes écrits en C fonctionnent en mode Texte. La fonction initgraph permet de passer en mode graphique. Ce mode graphique demeure jusqu'à la rencontre de le fonction closegraph qui arrêtera le mode graphique pour repasser en mode Texte. Et la fonction graphresult alors ? Elle sert uniquement à savoir si le passage du mode texte au mode graphique s'est bien passée.

3.1.3 Ma fonction d'initialisation du graphisme

Mais où est ce que l'on trouve cela … Bon, je vais vous livrer la potion magique du programmeur … Ça ne va pas non ! Bon un morceau …

La première phase consiste à chercher des exemples.

Pour trouver l'utilisation des fonctions on utilise les aides livrées avec le logiciel de programmation ou les livres comme par exemple "Aide mémoire de C" ou Internet.

Dans le cas présent, vous pouvez ouvrir le fichier Bgidemo.c qui se trouve dans le répertoire exemple et regarder. C'est bizarre il y a des ressemblances avec ce qui suit.

La seconde phase consiste à essayer de comprendre puis de faire des tests et enfin de se jeter à l'eau en passant au codage …

Et ça se passe toujours comme ça !

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

int Initialize(void)
{
  int    GraphDriver = DETECT;      /* Driver de carte graphique  */
  int    ErrorCode;                 /* Rapport des codes erreurs */
  int    GraphMode=0;

  initgraph( &GraphDriver, &GraphMode, "c:\\tc" );

  ErrorCode = graphresult();
  /* Lecture du résultat de l'initialisation */

  if( ErrorCode != grOk )
  {
      printf("Erreur de système graphique : %s\n",
            grapherrormsg(ErrorCode ));
```

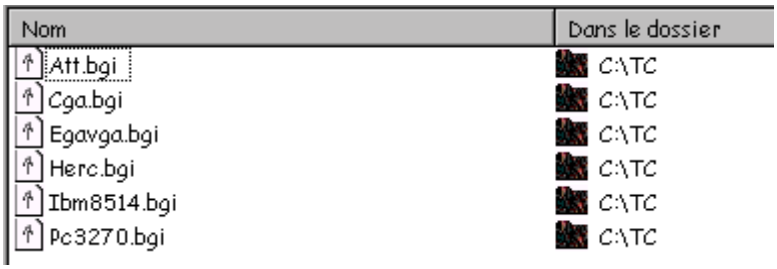
```
      return ( 1 );     /* Problème */
  }
}

int main ()
{
      if (Initialize ())      /* Ouverture du mode graphique */
      {
            printf ("Impossible de passer en mode graphique");
            return (1);
      }

      getch ();
      closegraph ();    /* Fermeture du mode graphique */
      return 0;
}
```

Exercice :

A l'aide de l'aide en ligne comprenez le programme (simple …).

3.1.4 Les fichiers BGI

Les fichiers BGI (*.BGI) sont des pilotes (non pas des pilotes de courses), mais des pilotes de carte vidéo qui permettent d'aller écrire dans la carte vidéo. Dans le turbo C vous trouverez différents pilotes. Par la fonction Windows et la commande rechercher, réaliser la recherche des fichiers *.BGI en partant du répertoire d'installation (c:\tc).



Je ne connais pas toutes les reliques d'écran et je ne veux pas connaître toutes les reliques d'écran. Je peux tout de même vous dire que :

- ☞ Egavga.bgi sert à l'EGA et le VGA, c'est celui que nous utiliserons.
- ☞ Herc.BGI est utilisée pour les écrans Hercule (noir et blanc).
- ☞ Cga.bgi pour les écrans CGA.

3.2 Un fichier squelette

Ras le bol de retaper toujours le même corps de programme …
Nous allons devoir faire beaucoup de petits programmes pour effectuer des tests sur les nouvelles fonctions. Je vous conseille donc de sauvegarder le précédent petit programme sous le nom squel.c (squelette) puis de le réutiliser pour chaque nouveau petit programme, en prenant soin de faire write as (sauvegarder sous) avec le nouveau nom de programme dès le début pour ne pas écraser squelet.c

3.3 Structure d'un programme

Voilà notre squelette de programme écrit mais en général, sans mode graphique, un programme C s'écrit avec le squelette suivant :

Déclaration des bibliothèques

```
#include <stdio.h>
#include <conio.h>
#include "prog.h"
```

Déclaration des variables globales : elles sont à proscrire impérativement.

```
int pas_bien = 0 ;
```

Déclaration des fonctions

```
int toto ()
{
        printf ("Bonjour, cette fonction ne fait pas grand chose !\n") ;
}
```

Fonction principale

```
int main ()
{
        toto () ;
}
```

3.4 Déclaration des bibliothèques

#include "… .h" indique que nous allons utiliser un fichier .h qui se trouve dans notre répertoire.
#include <… .h> indique que nous allons utiliser un fichier .h qui se trouve dans les librairies du C. Cet endroit est spécifié dans le menu Options Compiler.

Le sujet est vaste et je ne m'attarderai pas dessus.

3.5 Utilisation du graphisme

3.5.1 Ecriture de texte

La fonction à utiliser pour écrire du texte en mode graphique est **outtextxy**.
Par la fonction Ctrl-F1, j'obtiens son prototypage :

void far **outtextxy** (int x, int y, char far* textstring);

Exercices

1. Tester la fonction.

Exemple

J'ouvre mon squelette et je rajoute ce qui est en gras

```
int main ()
{
      if (Initialize ())     /* Ouverture du mode graphique */
      {
            printf ("Impossible de passer en mode graphique");
            return (1);
```

```
        }

        outtextxy (30,30, "Coucou c'est moi !");

        getch ();
        closegraph ();
        return (0);
}
```

Regardez, comprenez …

2. Ecrivez en 20,50, Oh C mon beau C, que C beau.

3.5.2 Notations

Vous êtes arrivé maintenant à un stade auquel le C ne doit plus vous effrayer, je vous décrits donc les fonctions comme elles le sont dans l'aide en ligne. Je vous rappelle que :

- **void** signifie indéfini. void à l'entrée d'une fonction signifie donc qu'elle ne renverra rien.
- **far** signifie loin. Une antiquité, faîtes comme si elle n'existait pas.

3.5.3 Couleur de tracé

La définition de la couleur de tracé est obtenue par la fonction :

void far setcolor (int color);

3.5.4 Lignes, cercles et rectangles

Les fonctions respectives pour dessiner une ligne, un cercle ou un carré sont :

void far line (int x1, int y1, int x2, int y2);
void far circle (int x, int y, int radius);
void far rectangle (int left, int top, int right, int bottom);

Exercice :

Tester ces 3 fonctions dans un même programme.

3.5.5 Effacement de l'écran

Arghhh !!! Notre vielle fonction clrscr () ne marche plus … Pour effacer l'écran il faut maintenant utiliser la fonction graphique :

void far cleardevice (void);

3.6 Exercice complet et instructif …

Ecrire un programme qui propose les 6 choix suivants à l'utilisateur :

1. Dessiner une ligne
2. Dessiner un rectangle
3. Dessiner un cercle
4. Choisir la couleur

9. Sortir

qui saisisse le choix de l'utilisateur et fait afficher ce qui est écrit (dans les cas 1,2,3).

Correction des exercices du chapitre 3

Exercice 3.4.1

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

int Initialize(void)
{
  int    GraphDriver = DETECT;     /* Driver de carte graphique  */
  int    ErrorCode;                /* Rapport des codes erreurs */
  int    GraphMode=0;

      initgraph( &GraphDriver, &GraphMode, "c:\\tc" );

      ErrorCode = graphresult();
      /* Lecture du r,sultat de l'initialisation */

      if( ErrorCode != grOk )
      {
            printf("Erreur de système graphique : %s\n",
            grapherrormsg(ErrorCode ));

            return ( 1 );     /* Problème */
      }
}

int main ()
{
      if (Initialize ())      /* Ouverture du mode graphique */
      {
            printf ("Impossible de passer en mode graphique");
            return (1);
      }

      outtextxy (20,50, "Oh C mon beau C, que C beau");

      getch ();
      closegraph ();    /* Fermeture du mode graphique */

      return 0;
}
```

Exercice 3.4.3

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

int Initialize(void)
{
  int    GraphDriver = DETECT;     /* Driver de carte graphique  */
  int    ErrorCode;                /* Rapport des codes erreurs */
  int    GraphMode=0;
```

```
      initgraph( &GraphDriver, &GraphMode, "c:\\tc" );

      ErrorCode = graphresult();
      /* Lecture du résultat de l'initialisation */

      if( ErrorCode != grOk )
      {
            printf("Erreur de système graphique : %s\n",
            grapherrormsg(ErrorCode ));

            return ( 1 );     /* Problème */
      }
}

int main ()
{
      if (Initialize ())      /* Ouverture du mode graphique */
      {
            printf ("Impossible de passer en mode graphique");
            return (1);
      }

      setcolor (7);
      line (10,10,250,170);

      setcolor (3);
      circle (250,350,100);

      setcolor (4);
      rectangle (100,200,350,400);

      getch ();
      closegraph ();    /* Fermeture du mode graphique */

      return 0;
}
```

∫ *Exercice 3.5*

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

int Initialize(void)
{
  int    GraphDriver = DETECT;      /* Driver de carte graphique  */
  int    ErrorCode;                 /* Rapport des codes erreurs */
  int    GraphMode=0;

      initgraph( &GraphDriver, &GraphMode, "c:\\tc" );

      ErrorCode = graphresult();
      /* Lecture du r,sultat de l'initialisation */

      if( ErrorCode != grOk )
      {
            printf("Erreur de système graphique : %s\n",
            grapherrormsg(ErrorCode ));
```

```
            return ( 1 );     /* Problème */
      }
}

int main ()
{
      char car = ' ';
      int couleur = 1; /* On évite le 0 car noir sur fond noir !!!! */
      int sortie = 0;

      if (Initialize ())      /* Ouverture du mode graphique */
      {
            printf ("Impossible de passer en mode graphique");
            return (1);
      }

      do
      {
            /* Affichage du menu */
            cleardevice ();
            setcolor (couleur);
            outtextxy (0,0,"1. Dessiner une ligne.");
            outtextxy (0,10,"2. Dessiner un rectangle.");
            outtextxy (0,20,"3. Dessiner un cercle.");
            outtextxy (0,30,"4. Changer la couleur.");
            outtextxy (0,50,"9. Sortir.");

            car = getch ();
            switch (car)
            {
                  case '1':
                        cleardevice ();
                        line (100,100,250,370);
                        outtextxy (200,400, "Appuyez sur une touche.");
                        getch ();
                        break;

                  case '2':
                        cleardevice ();
                        rectangle (100,200,350,400);
                        outtextxy (200,400, "Appuyez sur une touche.");
                        getch ();
                        break;

                  case '3':
                        cleardevice ();
                        circle (150,250,100);
                        outtextxy (200,400, "Appuyez sur une touche.");
                        getch ();
                        break;

                  case '4':
                        couleur ++;
                        if (couleur > 15)
                              couleur = 1;
                        break;

                  case '9':
                        sortie = 1;
                        break;
```

```
            }
        }
        while (!sortie);

        closegraph ();    /* Fermeture du mode graphique */
        return 0;
}
```