Utilisation de Turbo Pascal 7.0

Graphismes VGA

Par haypo

Date de publication : 24 juin 2001

Dernière mise à jour : 14 février 2016

DÉBUTANT

Cette série de tutoriels est destinée à aider les débutants à prendre en main Turbo Pascal 7.0.

Au menu de cette partie : étude de A à Z du mode graphique VGA (320x200 pixels en 256 couleurs), étude théorique du fonctionnement, puis pratique (avec du code assembleur) et, enfin, description de mon unité EcranVGA avec toutes ses procédures.

Commentez ♪

I - Introduction....3
II - Les modes vidéo....3
III - La pratique....3
IV - Stop !....4
V - Lire/écrire dans la VRAM....4
VI - L'unité EcranVGA....4
VI-A - Liste des fonctions....4
VII - Pour finir....6
VIII - Remerciements....6

Copyright ® 2001-2016 haypo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

I - Introduction

Tout ce qui est graphisme passe obligatoirement par la carte vidéo, qui va créer une image qu'elle envoie au moniteur vidéo. Cette carte vidéo est composée d'une mémoire vidéo (**VRAM** en anglais : *Video RAM*), d'un processeur qui peut optimiser le tracé de certaines figures (en 2D ou en 3D) et nous permet de changer de mode vidéo, et d'un autre processeur qui crée un signal pour le moniteur vidéo. Ce qui nous intéresse c'est changer de mode vidéo et de pouvoir lire et écrire dans la VRAM.

II - Les modes vidéo

Le temps du mode texte est révolu (désolé). Maintenant, les images sont en millions de couleurs et les écrans supportent au moins les modes 800x600 pixels (un pixel est le plus petit point que le moniteur puisse afficher dans un mode vidéo précis. Avec 800 pixels de largeur, l'écran est découpé en 800 petits points indiscernables).

Mais commençons léger : le mode **VGA**. Celui-ci fut une révolution quand il apparut, car il permettait d'afficher 256 couleurs simultanément dans 320x200 pixels. La raison pour laquelle je vais l'étudier est qu'il est très facile à utiliser avec Turbo Pascal. La mémoire vidéo dans ce mode est linéaire, c'est-à-dire que l'offset 0 correspond au pixel 0, de même pour le pixel 100 et le dernier 63999. Ça vous paraît normal, mais il n'en a pas toujours été ainsi ! De plus, chaque pixel est codé dans un octet qui peut avoir une valeur comprise entre 0 et 255, ce qui correspond à notre palette de 256 couleurs.

III - La pratique

Du côté pratique, il va falloir farfouiller dans les profondeurs de votre PC. J'utilise beaucoup l'assembleur, d'ailleurs. C'est le langage le plus proche de la machine ; il est composé uniquement d'instructions que comprend le processeur : plus, fois, copier un bloc de mémoire, lire un bloc de mémoire et quelques autres très techniques, c'est tout ! Il est très utilisé pour sa rapidité ou pour accéder au *hardware* (de l'anglais signifiant la partie matérielle de votre PC : souris, carte vidéo, processeur). Passons. D'abord il faut lancer le mode VGA :

```
asm
  mov ax,0013h
  int 10h
end;
```

Et oui, ça déroute un max !

- « **asm** » signifie qu'on va entrer du code assembleur ;
- « **mov** » est une instruction de copie ;
- « **ax** » est un registre, une variable réservée qui est logée physiquement dans le cœur du processeur ;
- « **h** » est un suffixe qui signifie que le nombre qui précède est au format hexadécimal (en base 16 pour les mathématiciens : 0Ah = 10 en décimal, 0F = 15, 10 = 20, FF = 255) ;
- **« int** » appelle une interruption, un genre d'unité chargée en mémoire ; ici, c'est l'interruption 10h : celle de la vidéo ;
- **« end;** » marque la fin du bloc assembleur.

En fait, c'est très simple à comprendre. 1300h peut être découpé en deux parties : 13h et 00h. 00h est le numéro de fonction de l'interruption, ici changement de mode vidéo. 13h est le numéro du mode vidéo, ici notre mode VGA.

Pour revenir au mode texte :

```
asm
  mov ax,0003h
  int 10h
end;
```


Copyright ® 2001-2016 haypo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Ce qui change est uniquement le numéro du mode vidéo. 03h est le mode texte classique, 80x25 caractères en 16 couleurs.

IV - Stop !

Pour ceux qui ne veulent pas se casser la tête, j'ai créé une **unité** : « **EcranVGA** » (à télécharger avec l'**ensemble de mes unités**), qui vous simplifiera la vie. Vous y trouverez toutes les procédures utiles. « ModeVga256; » lance le mode VGA et « ModeTxt; » revient au mode texte. C'est beaucoup plus simple non ? Ouf.

V - Lire/écrire dans la VRAM

La mémoire vidéo est accessible à l'adresse (hexadécimale) : A000:offset, avec offset une valeur comprise entre 0 et FFFFh (63999 nous suffira). Pour calculer l'offset, il faut faire : Y*320 +X. De manière plus claire :

- écrire : « Mem[$A000: Y*320 +X] := Couleur; » ;
- lire : « Couleur := Mem[$A000: Y*320 +X]; ».

« Couleur » est une variable de type « byte » (valeur entre 0 et 255), sa valeur correspond à un code de couleur défini par la palette des couleurs. Exemple :

- 0 = noir ;
- 1 = bleu foncé ;
- 7 = gris ;
- 12=rouge ;
- 14 = jaune ;
- 15=blanc, etc.

Les 16 premières sont les mêmes que celles du mode texte.

Vous pouvez utiliser **mes procédures** « **PutPixel** » et « **GetPixel** » (*Put* = (anglais) poser, donc écrire ; *Get* = (anglais) prendre, lire). Elles s'utilisent comme cela :

```
PutPixel (X,Y, Couleur);
Couleur := GetPixel (X,Y);
```

VI - L'unité EcranVGA

Voilà, vous savez tout sur le mode VGA ! Pour vous éviter les mêmes galères que j'ai endurées pour créer des procédures comme « lire dix pixels », « tracer un point », « dessiner un rectangle », je vous propose d'utiliser mon unité.

Les coordonnées X,Y des premières procédures doivent être de type Word, et comprises entre 0 et 319 pour X, et 0 et 199 pour Y.

VI-A - Liste des fonctions

- **ModeTxt** : lance le mode texte ;
- **ModeVga256** : lance le mode VGA ;
- **PutPixel (X,Y,Coul)** : trace un point à la position **X,Y** dans la couleur **Coul** ;
- **GetPixel (X,Y)** : lit la valeur du point à la position **X,Y** ;
- **PutPixelOfs (Ofst: Word;Coul: Byte)** : trace un point à l'offset **Ofst** de la VRAM dans la couleur **Coul** ;

Copyright ® 2001-2016 haypo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

- **EcritBloc (X,Y,Long: Word; var Src)** : écrit un bloc de données (linéaire) à la position **X,Y** d'une longueur **Long**. **Src** est le bloc, souvent de type « Array[…] of byte *» ;*
- **EcritBlocOfs (Ofst,Long: Word; var Src)** : écrit un bloc de données (linéaire) à l'offset **Ofst** d'une longueur **Long**. **Src** est le bloc, souvent de type « Array[…] of byte » *;*
- **EcritZone (X,Y,Larg,Haut: Word; var Src)** : écrit une zone de données à la position **X,Y** ayant comme dimensions **Haut** en hauteur et **Larg** en largeur. **Src** est le bloc, souvent de type « Array[…] of byte » *;*
- **LitBloc (X,Y,Long: Word; var Dst)** : lit un bloc de données (linéaire) à la position **X,Y** d'une longueur **Long** et l'écrit dans **Dst** (qui doit avoir une taille d'au moins **Long** octets) ;
- **LitBlocOfs (Ofst,Long: Word; var Dst)** : lit un bloc de données (linéaire) à l'offset **Ofst** d'une longueur **Long** et l'écrit dans **Dst** (qui doit avoir une taille d'au moins **Long** octets) ;
- **LitZone (X,Y,Larg,Haut: Word; var Dst)** : lit une zone de données à la position **X,Y** ayant comme dimensions **Haut** en hauteur et **Larg** en largeur, et l'écrit dans **Dst** ;
- **RempliBloc (X,Y,Larg: Word; Coul: Byte)** : remplit un bloc, c'est-à-dire répète une couleur sur **Larg** pixels (je ne détaille plus les paramètres, c'est tout le temps la même chose) ;
- **procedure RempliBlocOfs (Ofst,Larg: Word; Coul: byte)** ;
- **procedure RempliZone (X,Y,Larg,Haut: Word; Coul: byte)** ;
- **procedure RempliZoneOfs (Ofst,Larg,Haut: Word;Coul: byte)** ;
- **Procedure EcriVga (X,Y: Integer; Txt: String; Coul,CoulF: Byte)** : écrit le texte **Txt** à la position **X,Y** dans la couleur **Coul** sur un fond **CoulF** ;
- **Procedure EcriTsp (X,Y: Integer; Txt: String; Coul: Byte)** : écrit le texte **Txt** à la position **X,Y** dans la couleur **Coul** sur fond transparent (ne dessine pas les pixels transparents) ;
- **procedure ColoriZone (X,Y,Larg,Haut: Word; CoulAvant,CoulApres: byte)** : colorie une zone, c'est-à-dire que si la couleur **CoulAvant** est trouvée, elle est remplacée par la couleur **CoulApres** ;
- **procedure DessineLigneV (X,Y,Haut: word; Coul: byte)** : dessine une ligne verticale à la position **X,Y** d'une hauteur **Haut** dans la couleur **Coul** ;
- **procedure DessineLigneH (X,Y,Larg: word; Coul: byte)** : dessine une ligne horizontale à la position **X,Y** d'une largeur **Larg** dans la couleur **Coul**.

Et ce n'est que le début. J'ai quand même mis *deux ans* (j'ai dormi un petit peu entre temps) pour concevoir cette unité comme elle est actuellement (du 17/02/1999 pour la version 1.0, au 10/11/2000 pour la version 1.4.1). Et dire que je vous la file gratos !

Pour la suite, ce sont mes nouvelles procédures graphiques qui permettent de faire quelque chose de concret : tracé de ligne, rectangle, disque, etc. Elles acceptent les coordonnées négatives et tronquent ce qui dépasse automatiquement. Je ne vous raconte pas le temps passé pour le tracé du cercle ou d'une simple ligne (le plus dur, surtout quand la ligne est coupée, vive les maths !) :

- **procedure DessinePoint (X,Y: Integer; Coul: byte)** : trace un point à la position **X,Y** dans la couleur **Coul** (comme PutPixel, mais ne trace pas si ça sort de l'écran) ;
- **procedure DessineLigne (X,Y,X2,Y2: Integer; Coul: byte)** : dessine une ligne depuis le point **X,Y** jusqu'au point **X2,Y2** dans la couleur **Coul**. Vu comme j'ai bossé dessus, les points peuvent être alignés, confondus, en dehors de l'écran, la ligne peut être verticale ou horizontale. Bon OK, j'arrête de frimer ;
- **procedure DessineRectangle (X,Y: Integer; Larg,Haut: word; Coul: Byte)** : trace un rectangle à la position **X,Y** d'une dimension **Larg** sur **Haut** et d'une couleur **Coul** ;
- **procedure DessineRectanglePlein (X,Y: Integer; Larg,Haut: word; Coul: Byte)** : trace un rectangle plein à la position **X,Y** d'une dimension **Larg** sur **Haut** et d'une couleur **Coul** ;
- **procedure DessineCercle (X,Y: Integer; Rayon: Word; Coul: Byte)** : trace un cercle à la position **X,Y** d'un rayon **Rayon** et d'une couleur **Coul** ;
- **procedure DessineDisque (X,Y: Integer; Rayon: Word; Coul: Byte)** : trace un disque à la position **X,Y** d'un rayon **Rayon** et d'une couleur **Coul**.

Maintenant, pour éviter le scintillement de l'écran pendant l'écriture dans la VRAM, je vous propose encore des procédures :

- **AttendEcran** : attend l'écran. Explication : le moniteur (écran) est rafraîchi 70 fois par seconde, et il trace l'image de haut en bas et de gauche à droite. Le « spot » s'interrompt à la fin d'une ligne puis revient à gauche (donc 200 fois dans le mode VGA) et une autre fois (plus longtemps) pour aller du coin bas droit au

Copyright ® 2001-2016 haypo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

coin haut-gauche. C'est seulement à ce moment que l'on peut modifier la VRAM, car sinon, l'image modifiée scintille légèrement durant la modification. Ce qui donne :

```
repeat
  AttendEcran;
  { Trace le dessin }
until TouchPresse;
```

- **ChangeDestEcran** : change la destination de toutes les procédures, c'est-à-dire qu'au lieu d'écrire directement dans la VRAM, on peut dériver l'écriture (et la lecture) dans un tampon, ce qui est très utilisé pour éviter le scintillement en complément de « AttendEcran ». Voyez mon programme « Bouge », par exemple, pour plus de détails : la procédure TamponEcran en est la clé ;
- **AfficheEcran** : si variable « Ecran » (modifiable par « ChangeDestEcran ») n'est pas dirigée vers la VRAM directement, copie le contenu du pointeur vers lequel elle pointe dans la VRAM ;
- **AffichePtr (Source: Pointer)** : affiche le contenu du pointeur **Source** à l'écran (le copie dans la VRAM) ;
- **Procedure CopiePtr (Source,Dest: Pointer)** : copie le contenu du pointeur **Src** (de type PtrImage, soit un pointeur sur un tampon à la dimension de l'écran VGA) dans le pointeur **Dest** ;
- **EffaceEcran (Coul: Byte)** : efface le contenu de l'écran avec la couleur **Coul** (le remplit de cette couleur) ;
- **EffacePtr (Dest: Pointer; Coul: Byte)** : efface le contenu du pointeur **Dest** avec la couleur **Coul** (le remplit de cette couleur).

VII - Pour finir

Si avec ça vous n'arrivez pas à faire quelque chose de génial, c'est que… je m'exprime mal. Contactez-moi pour m'insulter ou, accessoirement, me poser des questions complémentaires.

Si vous faites partie de ceux qui veulent aller plus loin, vous pouvez également étudier mes unités **EcranX** et **Vesa2**.

La première gère le mode X : mode spécial qui permet d'avoir quatre pages pour le mode VGA (et donc éviter le scintillement), mais sa mémoire est codée sur quatre plans de bits différents : ceux dont le reste de la division par 4 de l'offset est 0, 1, 2 et enfin 3.

Sinon l'unité **Vesa2** vous permet d'accéder aux modes en « haute définition » (ils vont se marrer, les hommes de 2047 qui vous trouver ça dans leur archive « Programmation des années 2000 ») : 640x480, 800x600, 1024x768, 1280x1024 ou encore 1600x1200 pixels en 8, 15, 16, 24 ou 32 bits/pixels. Les procédures sont à peu près les mêmes (niveau nom bien sûr : PutPixel8, PutPixel15… PutPixel32 et DessineLigne8/15/16/24/32, par exemple).

VIII - Remerciements

Merci à **Jacques Théry** pour ses corrections.

Copyright ® 2001-2016 haypo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.