Les Régions sous *Delphi*

Ce tutorial, va vous apprendre à utiliser les régions, utilisées principalement pour faire des "trous" dans les forms, mais aussi à plein d'autres choses plus ou moins utiles.

SOMMAIRE

PREMIERE PARTIE : THEORIE

I] Créer une région.

Le type principal est le type *HRGN*. Tout comme les *THandle* et les *HPalette*, ce type n'est en fait qu'une variable numérique, indexant la régions sur laquelle on travaille.

```
Var
  R1 : HRGN;
```

Une fois que nous avons déclaré ce type, il faut créer la région, en utilisant une des cinq fonctions principales de création, dont voici la liste :

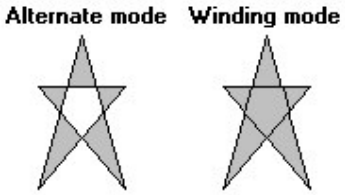
```
function CreateRectRgn(p1, p2, p3, p4 : integer) : HRGN;
ou
function CreateRectRgnIndirect(const p1 : TRect) : HRGN;
```
Ces deux fonctions créent une région rectangulaire.

```
function CreateEllipticRgn(p1, p2, p3, p4 : integer) : HRGN;
ou
function CreateEllipticRgnIndirect(const p1 : TRect) : HRGN;
```
Ces deux fonctions créent une région élliptique.

```
function CreateRoundRectRgn(r1, r2, r3, r4, r5, r6 : integer) : HRGN;
```
Cette fonction crée une région rectangulaire à bords arrondis. Les quatres premiers integer spécifient le rectangle de base et les deux derniers correspondent à la largeur et à la hauteur de l'éllipse utilisée pour les arrondis.

```
function CreatePolygonRgn(const Points ; Count : integer ; FillMode :
integer) : HRGN;
```
Cette fonction crée une région definit par un polygone fourni en paramètre. *Points* est un tableau de *TPoint*, *Count* spécifit le nombre de valeur du tableau et *FillMode* indique comment le polygone doit être remplit.

Les deux valeurs possibles de *FillMode* sont ALTERNATE et WINDING. Voici une illustration des deux (tiré de l'aide de windows SDK).




```
function CreatePolyPolygonRgn(const pPtStructs ; const pIntArray ; p3, p4 :
integer) : HRGN;
```
Cette fonction est la plus compliquée : elle permet de créer d'une seule traite plusieurs polygones. *pPtStructs* est un tableau contenant les points des polygones, *pIntArray* est un tableau contenant le nombre de points dans chaque polygones, *p3* specifit le nombre de polygones et *p4* comment le polygone doit être rempli (comme pour CreatePolygonRgn).

N'oublier pas de supprimer les variable de la mémoire lorsque vous n'en avez plus besoin, à l'aide de la méthode DeleteObject :

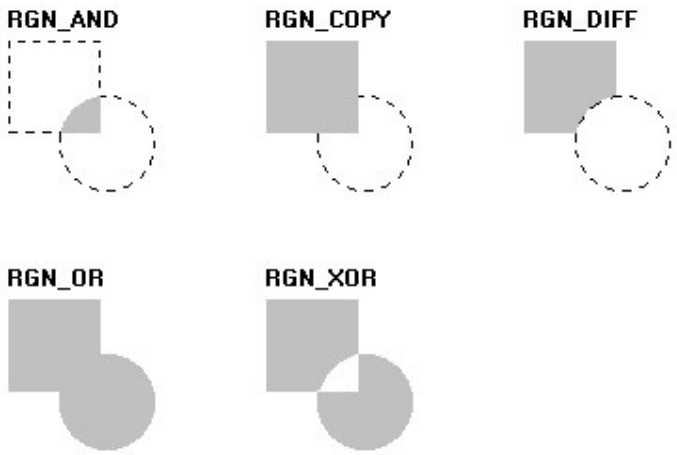
```
function DeleteObject(p1 : HGDIOBJ) : LongBool.
```

II] Combiner deux régions.

Combiner des régions est indispensable, pour inverser des régions, en assembler, en dissocier, ect...
Pour combiner des régions : il faut utiliser la function CombineRgn :

```
function CombineRgn(p1, p2, p3 : HRGN ; p4 : integer) : HRGN;
```

p1 correspond à la région de destination, *p2* et *p3* aux deux régions à combiner, et *p4* au type de combinaison.
Voici les différents types de combinaison possible illustrés (tiré de l'aide de windows SDK) :



III] Appliquer une région à un *handle*.

Appliquer une région à un handle se fait le plus simplement du monde, grace à la fonction SetWindowRgn :

```
function SetWindowRgn(hWnd : HWND ; hRgn : HRGN ; bRedraw : LongBool) :
integer;
```

hWnd correspond au handle sur lequel la région doit être appliqué, *hRgn* correspond à la région à appliquer et *bRedraw* definit si la région de la fenetre doit être redessinée à l'écran.

IV] Utiliser les régions pour faire du dessin.

Il existe plusieurs fonctions de dessin utilisant des régions : en voici quelques unes :

```
function FillRgn(DC : HDC ; hrgn : HRGN ; hbr : HBRUSH ) : LongBool;
```
Rempli la région *hrgn* du canvas spécifié par *DC* en utilisant le TBrush spécifié par *hbr*.

```
function PaintRgn(DC : HDC ; hrgn : HRGN) : LongBool;
```
Comme *FillRgn* sauf que la brush utilisée est celle du canvas spécifié.

```
function FrameRgn(DC : HDC ; hrgn : HRGN ; hbr : HBRUSH ; width, height
: integer ) : LongBool;
```
Dessine une bordure de la région *hrgn* du canvas spécifié par *DC*, en utilisant le TBrush spécifié par *hbr*. *width* et *height* spécifient la largeur et la hauteur de la bordure.

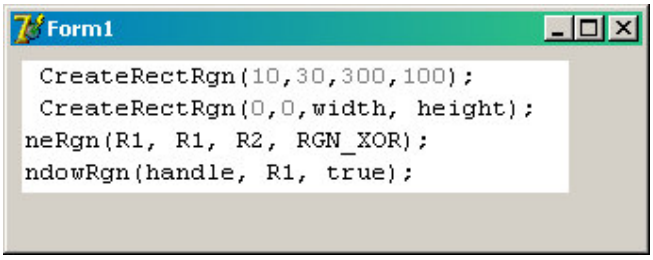
```
function InvRgn(DC : HDC ; p2 : HRGN) : LongBool;
```
Inverse les couleurs de la région *p2* du canvas spécifié par *DC*.

SECONDE PARTIE : PRATIQUE

I] Faire un trou dans une form :

Voici un petit bout de code pour trouer un rectangle dans une form :

```
procedure TForm1.FormCreate(Sender: TObject);
var
  R1, R2 : HRGN;
begin
  R1 := CreateRectRgn(10,30,300,100);
  R2 := CreateRectRgn(0,0,width, height);
  CombineRgn(R1, R1, R2, RGN_XOR);
  SetWindowRgn(handle, R1, true);
  DeleteObject(R1);
  DeleteObject(R2);
end;
```



II] Faire un trou complexe dans une form :

Nous allons creuser un smiley dans une form :

```
procedure TForm1.FormCreate(Sender: TObject);
var
  R1, R2, R3 : HRGN; //Declaration des HRGN
begin

  R1 := CreateRectRgn(0,0,width,height); //R1 englobe toute la form

  R3 := CreateEllipticRgn(81,81,81+75,81+81);//Le premier oeil
  CombineRgn(R1,R1,R3,RGN_XOR);//Soustraction du premier oeil de R1
  DeleteObject(R3);

  R3 := CreateEllipticRgn(229,81,229+75,81+81); //Le deuxieme oeil
  CombineRgn(R1,R1,R3,RGN_XOR);//Soustraction du deuxième oeil de R1
  DeleteObject(R3);

  R3 := CreateEllipticRgn(94, 194, 195+94, 133+194); //Une ellipse pour la
bouche
  CombineRgn(R1, R1, R3, RGN_XOR);//Soustraction de l'ellipse de R1
  DeleteObject(R3);

  R3 := CreateEllipticRgn(87,191, 87+213, 191+90);//Une ellipse à soustraire de
la bouche
  CombineRgn(R1, R1, R3, RGN_OR);//Ajout de l'ellipse à R1
  DeleteObject(R3);

  R2 := CreateEllipticRgn(25,38,25+333,38+311);//La tête
  CombineRgn(R1, R1, R2, RGN_XOR);//Creuse toute la tête sauf les régions
ajoutees precedemment
  SetWindowRgn(handle, R1, true);//Applique le tout à la form
  DeleteObject(R1);//Purge la mémoire
  DeleteObject(R2);//Purge la mémoire
end;
```

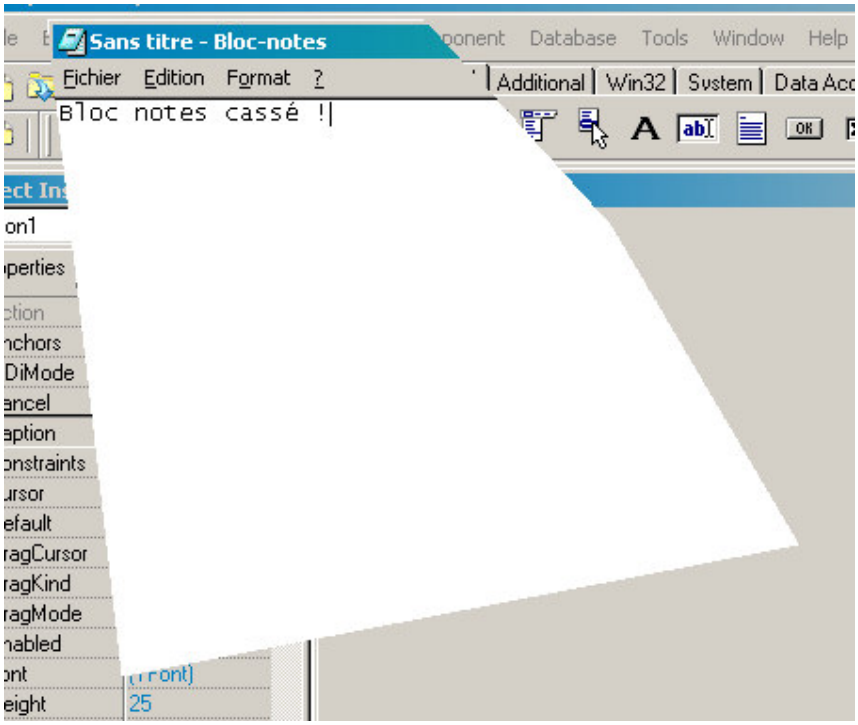
Voici le resultat, libre à vous d'utiliser le concept pour vos propres figures : les possibilités sont infinies.

III] Appliquer une région à une autre application :

Nous allons créer une région polygonale et l'appliquer à une instance du bloc-notes de windows (pour le lancer rapidement, aller dans démarrer\executer et tapez "notepad").

```
procedure TForm1.Button1Click(Sender: TObject);
var
  R1 : HRGN;
  NTPHandle : HWND; //Un handle
  ArrayOfPoint : array [0..5] of TPoint; //Un tableau de TPoint
begin
//Recuperation du handle du bloc-notes
  NTPHandle := FindWindow('Notepad',nil);
    if NTPHandle = 0 then //Si aucun bloc-notes n'est lancé
      MessageDlg('Pas de bloc notes lancé !', mtError, [mbOK], 0);
  ArrayOfPoint[0] := point(0,0); //Affectation des points
  ArrayOfPoint[1] := point(200,0);
  ArrayOfPoint[2] := point(300,110);
  ArrayOfPoint[3] := point(400,280);
  ArrayOfPoint[4] := point(40,350);
  ArrayOfPoint[5] := point(40,40);
//Creation d'une region polygonale
  R1 := CreatePolygonRgn(ArrayOfPoint, 5, WINDING);
//Application de la region au handle du bloc notes
  SetWindowRgn(NTPHandle, R1, true);
  DeleteObject(R1); //Suppresion de la region.
end;
```

Ce resultat n'est pas très intelligent mais c'est aussi un exemple de l'utilisation de la fonction *CreatePolygonRgn*.



IV] Le dessin à l'aide des régions :

Cet exemple montre comment créer une région polypolygonale mais aussi comment remplir, inverser les couleurs et dessiner les contours d'une région.

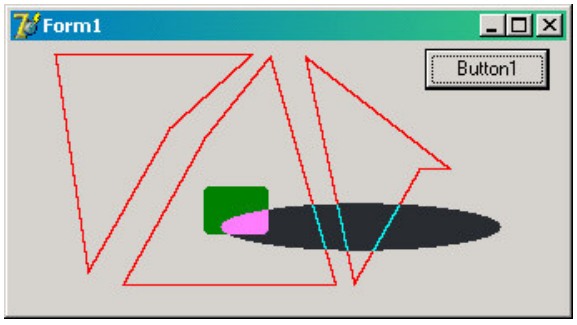
```
procedure TForm1.Button1Click(Sender: TObject);
var
  R1 : HRGN;
  R2 : HRGN;
  Points : array [0..11] of TPoint;
  PtCounts : array [0..2] of integer;
begin
//Les points du premier polygone
  Points[0] := point(30,30);
  Points[1] := point(150,30);
  Points[2] := point(100,75);
  Points[3] := point(50,163);
//Les points du deuxieme polygone
  Points[4] := point(160,30);
  Points[5] := point(120,80);
  Points[6] := point(70,170);
  Points[7] := point(200,170);
//Les points du troisième polygone
  Points[8] := point(180,30);
  Points[9] := point(210,170);
  Points[10] := point(250,100);
  Points[11] := point(270,100);
//Le nombre de points de chaque polygone
  PtCounts[0] := 4;
  PtCounts[1] := 4;
  PtCounts[2] := 4;

//Création du polypolygone
  R1 := CreatePolyPolygonRgn(Points, PtCounts, 3, WINDING);
  Canvas.Brush.Style := bssolid;
  Canvas.Brush.Color := clRed;
//Dessine les bordures de la région
  FrameRgn(GetWindowDC(handle),R1, Canvas.Brush.Handle, 1, 1);
  DeleteObject(R1);

//Création d'une région rectangulaire à bords doux
  R1 := CreateRoundRectRgn(120,110,160,140,5,5);
  Canvas.Brush.Color := clGreen;
//Remplisage de la région avec une belle couleur verte
  FillRgn(GetWindowDC(handle), r1, Canvas.Brush.Handle);
  DeleteObject(R1);

//Création d'une région elliptique
  R1 := CreateEllipticRgn(130,120,300,150);
//Inversion des couleurs de la région
  InvertRgn(GetWindowDC(handle), R1);
  DeleteObject(R1);
end;
```

Encore une fois, le résultat n'a pas grand interêt, mais il permet de mieux comprendre le fonctionnement de la fonction *CreatePolyPolygonRgn*.



TROISIEME PARTIE : COMPLEMENTS

Autres fonctions :

```
function PtInRegion(RGN : HRGN ; X, Y : integer) : longbool;
```
Renvoi *true* si le point définit par *X* et *Y* appartient à la région *RGN*.

```
function EqualRGN(p1, p2 : HRGN) : longbool;
```
Renvoi *true* si les deux régions passées en paramètres sont égales.

```
function GetRgnBox(RGN : HRGN ; var p2 : TRect) : integer;
```
Retrouve le rectangle "bordant" la région et l'assigne dans le TRect *p2*.

```
function OffsetRgn(RGN : HRGN ; XOffset, YOffset : integer) : integer;
```
Décale la région *RGN* de *XOffset* unités logiques vers la droite ou la gauche et de *YOffset* vers le haut ou le bas.

```
function GetWindowRgn(hWnd : HWND ; hRgn : HRGN) : integer;
```
Récupère la région associée au handle *hWnd* et l'assigne dans la région *hRgn*.

CREDITS \ LICENSE

L'auteur espère que vous avez apprécié son tutorial, et encore plus qu'il vous a servit à quelque chose.