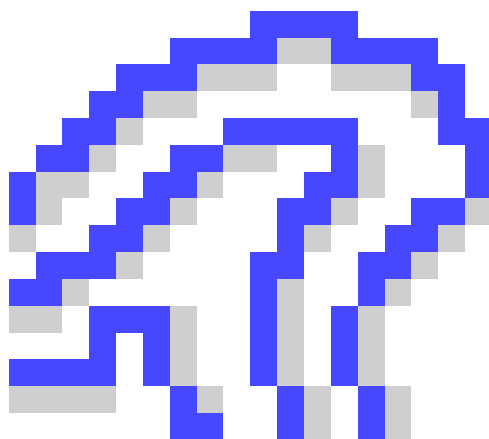


# 等值线控件 ContourOCX 5.52

## 用户手册



2007 年 9 月 5 日

**简介:**可将一系列点(规则或离散)生成等值线、等值面及对其切割及局部覆盖等。另外，增加不规则三角网的建立。

**注意:**

\*在测试版中，当数据点数超过 20 时，将会弹出 ABOUT 窗口

# Contour0CX 的成员函数

一、	等值线部分 .....	1
	1.规则网格点 .....	1
1. 1.	数据初始化 .....	1
1.2	计算、插补，并生成等值线 .....	2
	2.离散点生成等值线 .....	3
2.1	初始化 .....	3
2.2	加入离散点 .....	4
2.3	生成离散点等值线 .....	4
	3. 对等值线的操作 .....	5
	4.设置等值线属性 .....	5
二、	等值面的生成及操作 .....	6
	1、等值面的生成 .....	6
	2、等值面的属性 .....	7
	3、 多边形绘制操作 .....	8
三、	转化成矢量文件 .....	9
四、	图形的切割 .....	9
	1.切割初始化 .....	9
	2.添加边界 .....	9
	3.开始切割 .....	10
	4.切割后的等值线和等值面的属性 .....	11
4.1	切割后的等值线的属性 .....	11
4.2	切割后的等值面的属性 .....	11
	5. 切割后的绘制 .....	12
五、	等值线或等值面的部分屏蔽 .....	13
	1.屏蔽等值线或等值面 .....	13
	2. 绘制被屏蔽的等值线或等值面 .....	13
六、	生成不规则三角网（TIN） .....	14
	1 自然凸壳D-三角网 .....	14
	2 带强制边界的三角网 .....	15
	3 取得三角网的属性 .....	16
	4 由三角网得到等值线 .....	17
七、	生成矢量场 .....	17
	1 流线生成过程 .....	17
	2 矢量场属性取得 .....	19
	3 流线矢量场绘制 .....	20
	4 流线的部分屏蔽 .....	21
	5 箭头矢量场绘制 .....	21
八、	内存的释放 .....	22
九、	其它过程 .....	23
	1. 向屏幕输出图形时的重定位 .....	23

2. 设置颜色级 .....	23
3 使等值面透明 .....	24
4 输出插值后的数据 .....	24
5 使用DELPHI中出现异常的特别处理 .....	25
6 VB中MASK后颜色改变的问题 .....	26
十、联系方式 .....	27

## 一、等值线部分

### 1.规则网格点

#### 1. 1. 数据初始化

(1) `(1) void Initial(long row, long col, double step, short InterposeValue, short RestrictInterposedValue);`

功能：开辟内存空间

row:数据总行数

col:数据总列数

step:等值线间矩

InterposeValue:是否做插值

0:不做插值

1:做插值

RestrictInterposedValue:是否限制插值大小。

0: 不限制

1: 限制

(2) `void AddPoint(long row, long col, double x, double y, double value);`

功能：加入一个数据点

row:行坐标索引号(从 0 开始)

col:列坐标索引号(从 0 开始)

x:, y:坐标 (真实位置)

value:点的高程值

代码示例(C++):

```
float row=10;
float col=12;
float step=1;
CContourOCX m_sst;
.....
```

```

        m_sst.Initial(row,col,step);
        for(int i=0;i<row;i++)
        {
            for(int j=0;j<col;j++)
            {
                float randomValue(rand()%10+rand()%10/10.0);
                m_sst.AddPoint(i,j,j*20,i*20,randomValue);
            }
        }
    }
}

```

(3) `void AddCustomedStep(float x);`

功能：以特定值来绘制等值线。这个函数将使[void Initial(long row, long col, double step)]中的 step 失效。

## 1.2 .计算、插补，并生成等值线

有三种方法

(1) `void Calculate(long Compensate, long CompensateValue);`

参数：

Compensate:1-对缺失值进行插补；0-不插补(请确保数据阵列完整)

CompensateValue: 要插补的无效值

例如：

`m_ssst.Calculate(1, 9999);` //小于或等于 9999 的都会被认为无效值被插补。

或者

`m_ssst.Calculate(0, 9999);` //不进行插补

(2) `void Calculate2(long Compensate, long CompensateValue, float PointStep);`

参数：

Compensate:1-对缺失值进行插补；0-不插补(请确保数据阵列完整)

CompensateValue: 要插补的无效值

PointStep:值越小等值线越平滑，但速度会变慢。通常取 1

(3) `void Calculate3(long Compensate, long CompensateValue, int detail);`

参数：

Compensate:1-对缺失值进行插补；0-不插补(请确保数据阵列完整)

CompensateValue: 要插补的无效值

detail:值越大越平滑，同样也会降低速度

## 2.离散点生成等值线

### 2.1 初始化

(1) `void InitialRandomIIDW(long insertLineCount, long nearPointCount, float Step, long Smooth);`

功能: 用 IIDW(Inverse Distance Weighted).法进行初始化。取一定点数, 不考虑距离。

参数: insertLineCount:加密线数。大于 0。如为-1, 则以默认值 100 进行计算

NearPointCount:搜索最近点数。大于 0。如为-1, 则按默认值进行搜索

Step: 等值线间隔

Smooth:平滑参数。大于 0, 如为-1, 则按默认值 2 进行平滑

例如:

```
ContourOCX1.InitialRandomIIDW -1, -1, 1, -1
```

(2) `void InitialRandomIDW2(long insertLineCount, float distance,float DistPowerExponent, float Step, long smooth, long AtLeastOnePointFound, float ValueWhenNoFound);`

功能: 用 IDW ( Inverse Distance Weighted) .法进行初始化。取一定距离, 不考虑点数。

参数: insertLineCount:加密线数。大于 0。如为-1, 则以默认值 100 进行计算

distance:搜索半径。大于 0。

DistPowerExponent: 距离上的幂指数

Step: 等值线间隔

Smooth:平滑参数。大于 0, 如为-1, 则按默认值 2 进行平滑

AtLeastOnePointFound: 是否至少要找到一个点。

ValueWhenNoFound: 如果在所定半径内找不到点, 该点处的填充值。

例如:

```
ContourOCX1.InitialRandomIDW2 -1, 150, 2, linestep, Smooth, 0, 0
```

(3) `void InitialRandomCFWAI(long insertLineCount, float Step, long Smooth);`

功能: 用 CFWAI(Core Function Whole Area Interpolation).法进行初始化

参数: insertLineCount:加密线数。大于 0。如为-1, 则以默认值 100 进行计算

Step: 等值线间隔

Smooth:平滑参数。大于 0, 如为-1, 则按默认值 2 进行平滑

例如:

```
ContourOCX1.InitialRandomCFWAI -1, 1, -1
```

(4) `void InitialRandomCFPAI(long insertLineCount, long nearPointCount, float Step, long Smooth, float Power);`

功能: : 用 CFPAI(Core Function Part Area Interpolation).法进行初始化

参数:

insertLineCount:加密线数。大于 0。如为-1, 则以默认值 100 进行计算

nearPointCount: 搜索最近点数。大于 0。如为-1, 则按默认值进行搜索

Step: 等值线间隔

Smooth: 平滑参数。大于 0, 如为-1, 则按默认值 2 进行平滑.

Power: 距离上的幂指数

For example:

```
ContourOCX1.InitialRandomCFPAI -1, 30, 1, -1, -1
```

(5) `void InitialRandomKrigingOK(long insertLineCount, long nearPointCount, float Step, long Smooth, long ReservedValue);`

功能: 用 Ordinary Kriging.法进行初始化。采用线性模型, 块金值 (nugget) 设为 0

参数: insertLineCount:加密线数。大于 0。如为-1, 则以默认值 100 进行计算

NearPointCount:搜索最近点数。大于 0。如为-1, 则按默认值进行搜索

Step: 等值线间隔

Smooth:平滑参数。大于 0, 如为-1, 则按默认值 2 进行平滑

ReservedValue:保留值

例如:

ContourOCX1.InitialRandomKrigingOK -1, 30, 1, 2, -1

(6) `void InitialRandomNN(float Step, long Smooth, long para1);`

功能: 用 Nature Neighbor.法进行初始化。

参数:

Step: 等值线间隔

Smooth:平滑参数。大于 0, 如为-1, 则按默认值 2 进行平滑

para1:保留值

例如:

ContourOCX1.InitialRandomNN 1.0, 2, 1

## 2.2 加入离散点

有两种方式加入离散点

(1) `void AddPointRandom(double x, double y, double value);`

功能: 逐个加入离散点。

参数: x,y,z:坐标值和高程值

(2) `long AddRandomPointsFromFile(BSTR* path);`

功能:从文件中直接导入数据

参数: path:数据文件的路径。文件应为文本文件 (TXT), 分隔符为空格或其它, 分隔符的个数不限。例如:

x1 y1 value1	x1 ,y1,value1	x1, y1 value1
x2 y2 value2	x2, y2, value2	x2 y2,, value2
.....	或	.....
xn yn valuen	xn, yn, valuen	xn yn valuen

## 2.3 生成离散点等值线

`void CalculateRandom();`



功能 :生成离散点的等值线

### 3. 对等值线的操作

(1) `void GetLineCount(long* LineCount);`

功能: 得到等值线条数

参数:

\*LineCount:返回等值线条数

(2) `void GetLineValue(long LineIndex, float* value);`

功能: 得到等值线的值

参数:

LineIndex:等值线索引值

\*value:线的值

(3) `void GetCtrlPointCount(long LineIndex, long*PointCount);`

功能: 得到等值线的中控制点的个数

参数:

LineIndex:等值线索引值

\*PointCount: 控制点的个数

(4) `void GetCtrlPoint(long LineIndex, long PointIndex, double*X, double*Y, double*Value);`

功能: 得到等值线中某个控制点的属性

参数:

LineIndex:等值线索引值

PointIndex:控制点索引值

\*X, \*Y, \*Value: 坐标和等值线值

### 4.设置等值线属性

(1) `void SetLineSize(long LineIndex, short LineSize);`

功能:设置线的宽度

参数:

LineIndex: 线的索引值

LineSize:线的宽度

(2) `void SetLineColor(long LineIndex, long LineColor);`

功能:设置线的颜色

参数:

LineIndex: 线的索引值

LineColor:线的颜色.

(3) `void ResetContoursColor();`

功能:设置所有线的颜色

parameter(s):none

例子: ( VB):

---

```
' change color from red to green, then from green to blue
ContourOCX1.ClearColorClass
```

```
ContourOCX1.AddNewElementToColorClass RGB(255, 0, 0)
ContourOCX1.AddNewElementToColorClass RGB(0, 255, 0)
ContourOCX1.AddNewElementToColorClass RGB(0, 0, 255)
ContourOCX1.AddNewElementToColorClass RGB(255, 0, 255)
' ContourOCX1.SetDefaultZValueRange 1, 10
ContourOCX1.ResetContoursColor
```

---

(4) `void ResetContourPosition();`

功能:重置所有等值线的位置

参数:无

For Example:

---

```
ContourOCX1.SetDefaultPositionValus Picture1.Width, Picture1.Height, 0, 500, 0,
500
ContourOCX1.ResetContourPosition
ContourOCX1.DrawContours Picture1.hDC
```

---

(5) `void DrawContours(long dc);`

Function: 绘制所有等值线。

参数:

dc: 设备环境句柄

## 二、等值面的生成及操作

在等值面生成前，请确保等值线的生成已完成。

### 1、等值面的生成

(1) `void ConvertToPolygon(long* Succeed);`

功能: 将等值线转化为等值面

参数:

Succeed:成功与否。大于 0 成功，小于等于 0 失败

## 2、等值面的属性

(1) `void GetPolygonCount(long* PolygonCount);`

功能：得到等值面个数

参数：

\*PolygonCount:返回等值面个数

(2) `void GetPolygonPointCountValueArea(long index, long* PointCount, float* MinValue, float* MaxValue, float* AreaWithoutHoles, float* AreaWithHoles);`

功能：得到一个等值面的多边形的控制点数，值范围 (MinValue, MaxValue) 和面积 (AreaWithoutHoles, AreaWithHoles);

参数：

index:等值面多边形的索引号

\*PointCount:控制点个数

\*MinValue、\*MaxValue:等多边形所代表的值范围

AreaWithoutHoles、AreaWithHoles: 多边形的面积。对于不包含其它子多边形的多边形来说 AreaWithoutHoles 和 AreaWithHoles 值是相等的。一般来说，其实际面积为 AreaWithHoles。

有以下公式成立： $\text{AreaWithHoles} = \text{AreaWithoutHoles} - \text{所包含的子多边形面积}$

(3) `void GetPolygonPointXY(long polygonIndex, long pointIndex, float* X, float* Y);`

功能：得到多边形的一个控制点

参数：

polygonIndex:多边形的索引号

pointIndex: 控制点索引号

\* X, \* Y: 实际坐标

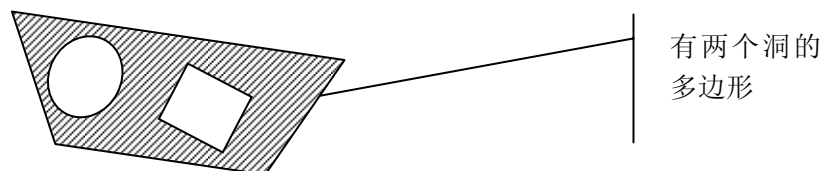
(3) `void GetPolygonHoleCount(long polygonIndex, long* holeCount);`

功能：得到多边形中洞的数目

参数：

polygonIndex:多边形的索引号

\*holeCount: 洞的数目



(4) `void GetPolygonIndexFromHoleIndex(long polygonIndex, long holeIndex, long* PolygonIndexInList);`

功能：洞作为一个多边形，其存储位置的索引号

参数:

    polygonIndex: 多边形的索引号

    holeIndex: 洞在包含它的多边形中的序号

    \* PolygonIndexInList: 洞作为一个多边形的序号

### 3、 多边形绘制操作

(1) **void ResetPolyPosition ();**

    功能: 根据实际坐标重新定位屏幕上的显示位置。

(2) **void ResetOnePolygonColor(long polygonIndex, long fillColor, long borderColor);**

    功能: 改变多边形显示时的颜色属性

    参数: polygonIndex: 多边形的索引号

        fillColor: 多边形的填充颜色

        borderColor: 多边形的边界颜色

(3) **void ResetPolygonsColor(long BorderColor);**

    功能: 设置所有等值面的颜色

    参数:

        BorderColor : 等值面的边界颜色.

    For example( in VB):

```
-----  
' change color from red to green, then from green to blue  
ContourOCX1.ClearColorClass  
ContourOCX1.AddNewElementToColorClass RGB(255, 0, 0)  
ContourOCX1.AddNewElementToColorClass RGB(0, 255, 0)  
ContourOCX1.AddNewElementToColorClass RGB(0, 0, 255)  
ContourOCX1.AddNewElementToColorClass RGB(255, 0, 255)  
' ContourOCX1.SetDefaultZValueRange 1, 10  
ContourOCX1.ResetPolygonsColor RGB(255, 255, 0)  
-----
```

(3) **void DrawAllPolygons(long dc);**

    功能: 绘制所有多边形到指定的设备环境变量上。

    参数: dc:设备环境句柄

    VB 代码示例: ContourOCX1.DrawAllPolygons Picture2.hDC

## 三、转化成矢量文件

(1) void InitializeSHPFile(BSTR\* Path, BSTR\* SHPType);

功能::初始化文件路径及矢量类型

参数:

Path:保存路径

SHPType:矢量类型. 参数可选"line"和 "polygon"., 分别生成等值线和等值面。

(2) void AddAppendedField(BSTR\* ShpType,BSTR\*FieldName,BSTR\*FieldType,long CharLength);

功能: 向 DBF 中添加附加的字段

参数:

SHPTYPE:Shape type.

FieldName: 矢量类型. 参数可选"line"和 "polygon"., 分别针对等值线和等值面矢量文件。

FieldType: 字段类型. 包括"integer","float" 和 "string"三个可选常量。

CharLength:数据长度。 对字符型字段来说, 它等于字符串长度; 对整型字段来说, 它是整数的数据位数; 对于浮点型字段, 它等于整数位数与小数位数的总和 (整数占 2/3,小数占 1/3)。

(3) void CreateShapeFile();

功能 :生成矢量文件

注: 在没有执行 AddAppendedField 的情况下, 生成的矢量文件 (等值线或等值面) 中的 DBF 文件都包含三个字段:

①MinValue: 浮点型, 表示等值线 (面) 的开始值;

② MaxValue: 浮点型, 表示等值线 (面) 的结束值; 如果是等值线, MinValue=MaxValue.

③IDNo: 整型, 表示等值线 (面) 的索引号。

## 四、图形的切割

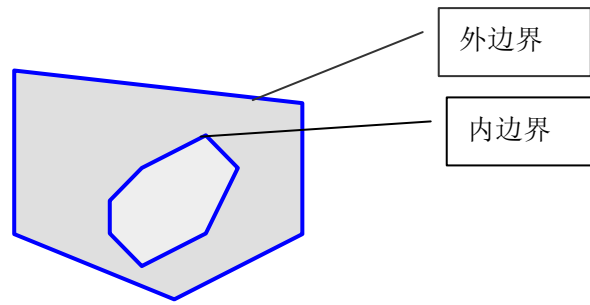
### 1.切割初始化

void InitializeForCut();

功能 : 切割之前的初始化

### 2.添加边界

边界有内边界(hole)和外边界 (border) 两类。如图所示



(1) `void AddNewFrame(BSTR* BorderOrHole);`

功能：开始添加一个新边界

参数：\* BorderOrHole: 边界种类 ("border" 或 "hole")

(2) `void AddFrmCtrlPoint(double x, double y);`

功能：添加边界上的一个顶点

参数：

x,y:顶点坐标

例如 (VB):

*'circle border*

*s = "border"*

**Dim i As Integer**

**Dim angle As Double**

**ContourOCX1.AddNewFrame s**

**For i = 1 To 71**

**angle = 3.1415926 \* 2 / 72 \* i**

**ContourOCX1.AddFrmCtrlPoint 110 + Cos(angle) \* 70, 120 + Sin(angle) \* 100**

**Next**

*'add a hole in the circle area*

**s = "hole"**

**ContourOCX1.AddNewFrame s**

**For i = 1 To 71**

**angle = 3.1415926 \* 2 / 72 \* i**

**ContourOCX1.AddFrmCtrlPoint 120 + Cos(angle) \* 50, 120 + Sin(angle) \* 50**

**Next**

### 3.开始切割

对当前已存在的等值线或等值面进行切割

(1) `void CutLinesAlongFrame();`

功能：切割等值线

参数：无

(2) `void CutPolygonAlongFrame();`

功能：切割等值面

参数：无

## 4.切割后的等值线和等值面的属性

### 4.1 切割后的等值线的属性

(1) `void GetLineCountAfterCut(long* LineCount);`

功能：得到切割后的等值线条数

参数：

\*LineCount:返回等值线条数

(2) `void GetCtrlPntCountAndValueAfterCut(long LineIndex, long* PointCount,double*Value);`

功能：得到切割后的等值线中的控制点的个数和等值线值

参数：

LineIndex:等值线索引值

\*PointCount: 控制点的个数

\*Value: 等值线值

(3) `void GetCtrlPointAfterCut(long LineIndex, long PointIndex, double* x, double* y);`

功能：得到切割后的等值线中的某个控制点的属性

参数：

LineIndex:等值线索引值

PointIndex:控制点索引值

\*X, \*Y,\*Value: 坐标和等值线值

### 4.2 切割后的等值面的属性

(1) `void GetPolygonCountAfterCut(long* PolygonCount);`

功能：得到切割后的等值面个数

参数：

\*PolygonCount:返回等值面个数

(2) `void GetPolygonPntCountValueAfterCut (long polyIndex, long* PointCount, float* MinValue, float* MaxValue);`

功能：得到一个切割后的多边形的控制点数，值范围(MinValue, MaxValue)

参数：

index:等值面多边形的索引号

\*PointCount:控制点个数

\*MinValue、\*MaxValue:多边形所代表的值范围

(3) `void GetPolygonPointXYAfterCut(long polygonIndex, long pointIndex, float * X, float * Y);`

功能：得到切割后的多边形的一个控制点属性

参数：

    polygonIndex: 多边形的索引号

    pointIndex: 控制点索引号

    \* X, \* Y: 实际坐标

## 5. 切割后的绘制

(1) `void ResetContourPositionAfterCut();`

功能: 剪切后等值线的重新定位

参数: 无

(2) `void SetLineColorAfterCut(long LineIndex, long LineColor);`

功能: 给切割后的等值线设置颜色

参数：

    LineIndex: 线的索引号

    LineColor: 线的颜色.

(3) `void ResetContoursColorAfterCut();`

功能: 设置所有切割后的等值线的颜色

参数: 无

(4) `void DrawContoursAfterCut(long dc);`

    功能: 绘制所有切割后的等值线

    参数：

    dc: 设备环境句柄

(5) `void ResetPolyPostionAfterCut();`

    功能： 重新设置所有等值面的位置

    参数： 无

( 6 ) `void ResetOnePolygonColorAfterCut(long polygonIndex, long fillColor, long borderColor);`

    功能： 改变切割后的多边形颜色属性

    参数: polygonIndex: 多边形的索引号

        fillColor: 多边形的填充颜色

        borderColor: 多边形的边界颜色

(7) `void ResetPolygonsColorAfterCut(long BorderColor);`

    功能: 设置所有切割后的多边形的颜色

    参数：

        BorderColor : 边界颜色



(8)void DrawAllPolygonsAfterCut(long dc);

功能：绘制切割后的所有多边形到指定的设备环境变量上。

参数：dc:设备环境句柄

## 五、等值线或等值面的部分屏蔽

### 1.屏蔽等值线或等值面

(1) void InitializeForMask();

功能：屏蔽前初始化

(2)void MaskLinesByFrame(long dc,long width,long height,float x1,float x2,float y1,float y2);

功能：部分屏蔽等值线

参数：

dc: 设备环境句柄

width,height: 绘图窗口的宽和高

x1,x2,y1,y2: 实际坐标范围

(3)void MaskPolygonsByFrame(long dc,long width,long height,float x1,float x2,float y1,float y2);

功能：部分屏蔽等值线

参数：

dc: 设备环境句柄

width,height: 绘图窗口的宽和高

x1,x2,y1,y2: 实际坐标范围

例如 (VB)

```
'-----  
ContourOCX1.InitializeForMask  
Call addFrames 'add frames  
Dim lineCount As Long, pointCount As Long, Value As Single, color As Variant  
ContourOCX1.GetLineCount lineCount  
...  
ContourOCX1.MaskLinesByFrame Picture2.hDC, Picture2.Width, Picture2.Height,  
10, 500, 0, 500  
ContourOCX1.DrawMaskedLinesOrPolygons Picture2.hDC  
'-----
```

### 2. 绘制被屏蔽的等值线或等值面

(1) void DrawMaskedLines(long dc);

功能:绘制所有经屏蔽的等值线

参数：

dc: 设备环境句柄

(2) `void DrawMaskedPolygons(long dc);`

功能: 绘制所有经屏蔽的等值面

参数:

dc: 设备环境句柄

(3) `void DrawMaskedLinesOrPolygons(long dc);`

功能: 绘制当前被部分屏蔽的等值线或等值面

参数:

dc: 设备环境句柄

例如:

VB:

Picture2.Cls

ContourOCX1.DrawMaskedLinesOrPolygons Picture2.hDC

VC:

HDC dc=::GetDC(m\_Picture2.m\_hWnd );

ContourOCX1.DrawMaskedLinesOrPolygons((long)dc);

::ReleaseDC(m\_Picture2.m\_hWnd,dc); //release dc

Delphi:

dc:=GetDC(Picture2.Handle);

ContourOCX1.DrawMaskedLinesOrPolygons(dc);

ReleaseDC(Picture2.Handle,dc);

## 六、生成不规则三角网 (TIN)

### 1 自然凸壳 D-三角网

(1) `void InitialRandom (float Step) ;`

功能: 初始化

参数: step 保留

(2) `void DrawTIN(long dc);`

功能: 将三角网画在设备变量上

参数:

dc: 环境变量

例如:

-----  
ContourOCX1.InitialRandomTIN 1

Call AddRandomPoint

ContourOCX1.CalculateRandom

dc = GetDC(Picture1.hwnd)

ContourOCX1.DrawTIN dc  
ReleaseDC Picture1.hwnd, dc

## 2 带强制边界的三角网

(1) `void InitialRandomTINconstrain(long deleteWhenOutOfBorder);`

功能: 初始化

参数:

`deleteWhenOutOfBorder` 处于约束边界以外的三角形是否去除

1—去除

0—保留

(2) 加入边界的约束点, 有两种方法

a. `void AddFrmCtrlPointForTIN(double x, double y, double value);`

功能: 逐点加入

参数: x, y: 坐标 (真实位置)

value: 点的高程值

例如:

```
ContourOCX1.InitializeForMask
Dim s As String
s = "border"
ContourOCX1.AddNewFrame s
ContourOCX1.AddFrmCtrlPointForTIN 120, -40, 2
ContourOCX1.AddFrmCtrlPointForTIN 128, -30, 3
ContourOCX1.AddFrmCtrlPointForTIN 140, -41, 4
```

b. `void AddFrmCtrlPointFromTXTFile(BSTR path);`

功能: 从文本文件加入 (内外边界的定义参看 图形切割)

参数:

path: 文件路径名

文件的格式如下:

```
border
136.60 44.93 0.0
134.12 43.25 0.0
136.60 36.89 0.0
136.60 38.89 0.0
136.60 40.89 0.0
136.60 44.72 0.0
hole
133.56 34.02 0.0
133.07 33.90 0.0
132.94 34.11 0.0
127.60 23.81 0.0
```

外边界开始标志

外边界数据

内边界开始标志

内边界数据

X 坐标

Y 坐标

Z 值

例如:

```
path = App.path + "\\Australia.txt"
ContourOCX1.AddFrmCtrlPointFromTXTFile path
```

### 3 取得三角网的属性

```
(1) void GetTriangleCount(long* count);
```

功能: 得到三角网中三角形个数

参数:

count: 三角形个数

```
(2) void GetTriangleEdgeCount(long* EdgeCount);
```

功能: 得到三角网中所有边的条数

参数:

EdgeCount: 边的条数

```
(3) void GetTrianglePoint(long TriangleIndex, long PointIndex, float* x, float* y, float* value);
```

功能: 得到三角形顶点

参数:

TriangleIndex: 三角形的索引号

PointIndex: 顶索引号 (小于 3)

x, y, value: 顶点位置 and 值

```
(4) void GetTriangleEdges(long TriangleIndex, long* EdgeIndex1, long* EdgeIndex2, long* EdgeIndex3);
```

功能: 得到一个三角形的三条边的索引值

参数:

TriangleIndex: 三角形的索引值

EdgeIndex1, EdgeIndex2, EdgeIndex3: 三条边的索引值

```
(5) void GetTriangleEdgeEndPoints(long EdgeIndex, float* x1, float* y1, float* x2, float* y2);
```

功能: 得到一条边的两个端点

参数: EdgeIndex: 边的索引值

x1, y1: 一个端点的坐标

x2, y2: 另一个端点的坐标

```
(6) void GetTriangleNeighboursViaEdge(long EdgeIndex, long* TriangleIndex1, long* TriangleIndex2);
```

功能: 得到共用一条边的两个三角形

参数: EdgeIndex: 边的索引值

TriangleIndex1, TriangleIndex2: 两个三角形的索引值。如果为 -1, 则表示不存在

```
(7) void GetTriangleNeighborsViaSelf(long SelfIndex, long* TriangleIndex1, long* TriangleIndex2, long* TriangleIndex3);
```

功能：得到一个三角形的三个邻居

参数：SelfIndex:三角形的索引值

TriangleIndex1, TriangleIndex2, TriangleIndex3: 三个相邻三角形的索引值

## 4 由三角网得到等值线

```
void TIN2Contours(float SimpleStep, BSTR CustomizeSteps);
```

功能：由三角网生成等值线

参数：SimpleStep: 固定步长值。间隔不变。

CustomizeSteps: 自定义的，间隔有变化。如“1 5 3.5”等。

## 七、生成矢量场

### 1 流线生成过程

```
(1) void InitialVector2D(long insertLineCount, long nearPointCount, float  
IDW2Dist, float DeltaH, long RKType, long SpeadLineDensity, long  
DistBetweenLines, long DistBetweenArrows, BSTR InterpolateMethod);
```

功能：生成流线前的初始化

参数：

insertLineCount:加密线数。大于0。如为-1，则以默认值100进行计算

NearPointCount:搜索最近点数。大于0。如为-1，则按默认值进行搜索

IDW2Dist: 使用方法IDW2时搜索的最小半径

DeltaH: 流线增长最小单位。如取-1，取系统默认值

RKType:保留值

SpeadLineDensity:流线生成密度。值越大，流线越密。

DistBetweenLines: 流线取点间隔。值越小，流线越密。

DistBetweenArrows: 流线上箭头间隔。以DeltaH为单位。

InterpolateMethod: 插值方法。(IDW1,IDW2,Kriging,CFWAI,CFPAI,NN)

例如：

```
ContourOCX1.InitialVector2D -1, 30, 20, -1, -1, 30, 1, 60, "kiriging"
```

```
(2) void AddCustomedVectorLnSeed(float x, float y);
```

功能：定制流线。

参数：x,y:生成流线的种子

例如：

```
ContourOCX1.InitialVector2D -1, 30, 20, -1, -1, CInt(LineDensityEdit.Text),  
CInt(distBetweenLinesEdit.Text), CInt(DistBtwnArrowsEdit.Text), m
```

```

ContourOCX1.AddCustomedVectorLnSeed 30, 20
ContourOCX1.AddCustomedVectorLnSeed 40, 80
ContourOCX1.AddCustomedVectorLnSeed 140, 180
Dim i As Integer
    Dim X As Double, Y As Double, u As Double, v As Double

If DataSourceIndex = 0 Then
    Call AddRandomPoints
Else
    Dim path As String
    'path = App.path + "\"vectorPoint.txt"
    path = App.path + "\"out3.txt"

    Dim count As Integer
    count = ContourOCX1.AddRandomVectorPointsFromFile(path)
End If

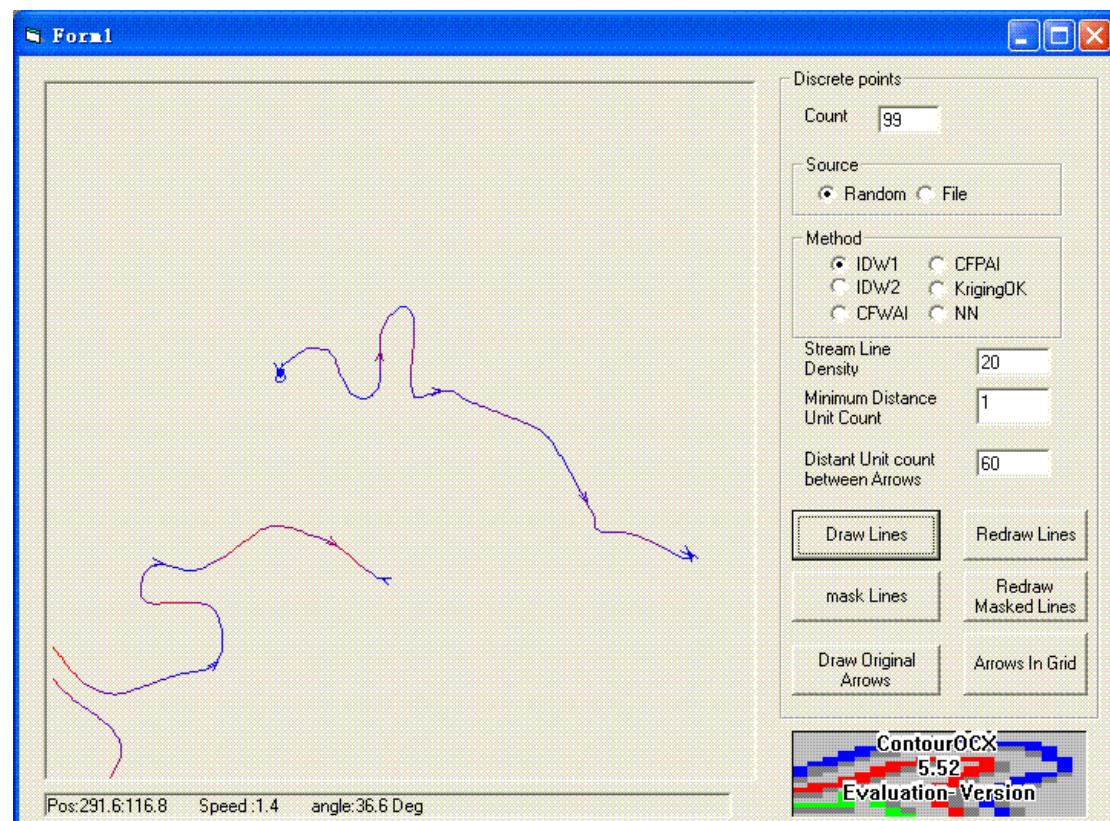
```

```

ContourOCX1.CalculateVector
Call SetStreamLinesColor
ContourOCX1.ResetVector2DLinePosition
ContourOCX1.DrawVectorLines Picture1.hdc

```

生成的图像为：



```
(3) void AddVectorPointRandom(double x, double y, double uValue, double vValue);
```

功能：逐个添加矢量点

参数：x, y:位置

u,v:速度的 x 和 y 向的分量

```
(4) long AddRandomVectorPointsFromFile(BSTR* path);
```

功能:从文件中直接导入数据

参数: path:数据文件的路径。文件应为文本文件（TXT），分隔符为空格或其它，分隔符的个数不限。例如：

x1 y1 u1 v1		x1, y1, u1, v1
x2 y2 u2 v2		x2 ,y2 ,u2 ,v2
.....	或	.....
xn yn un,vn		xn ,yn ,un,,vn

返回值为读入的记录数

```
(5) void CalculateVector();
```

功能：计算流线

参数：无

## 2 矢量场属性取得

```
(1) void GetVectorLineCount(long* lineCount);
```

功能：得到流线条数

参数：lineCount:流线条数

```
(2) void GetVectorLinePointCount(long lineIndex, long* pointCount);
```

功能：得到一条流线上的控制点数

参数：lineIndex：流线索引号

pointCount:控制点数

```
(3) void GetVectorLinePointPosAndValue(long lineIndex, long pointIndex, float* x, float* y, float* u, float* v);
```

功能：得到一条流线上的控制点的属性

参数：

lineIndex：流线索引号

pointIndex:控制点索引号

x,y:位置

u,v:控制点上矢量在 X 方向和 Y 方向的分量

```
(4) void GetVectorPointValues(float x, float y, float* u, float* v, float* angle);
```

功能：得到任意一点上的矢量属性

参数：

x,y:位置  
u,v:控制点上矢量在 X 方向和 Y 方向的分量  
angle:矢量的方向

### 3 流线矢量场绘制

```
(1) void ResetVector2DLinePosition();
```

功能：重新定位流线在屏幕上的位置。必须与 SetDefaultPositionValus 配合使用。

参数：无

```
(2) void ResetVectorLinesColor();
```

功能：更新流线颜色。

参数：无

```
(3) void DrawVectorLines(long dc);
```

功能：画流线

参数：dc:设备的 dc

例如：

.....

ContourOCX1.CalculateVector

ContourOCX1.ClearColorClass

ContourOCX1.AddNewElementToColorClass RGB(0, 0, 255)

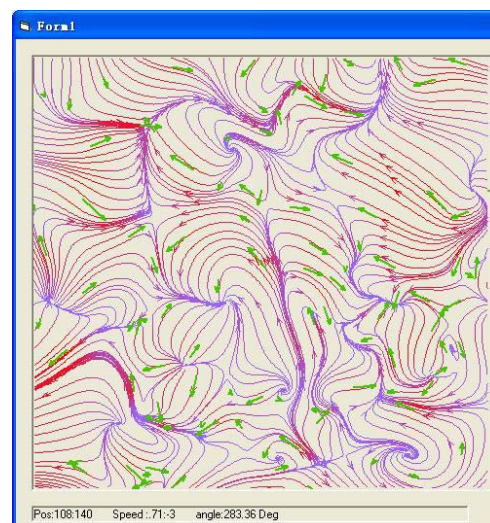
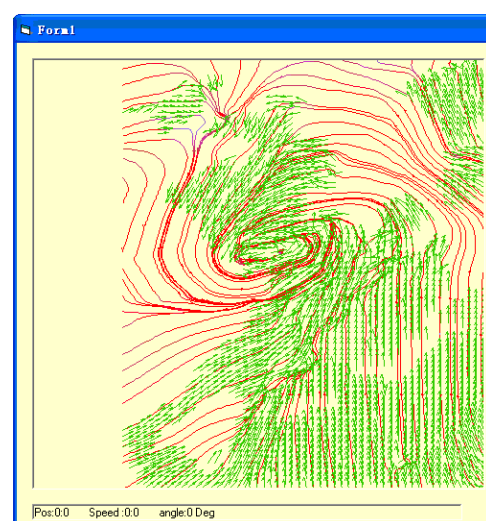
ContourOCX1.AddNewElementToColorClass RGB(0, 255, 0)

ContourOCX1.AddNewElementToColorClass RGB(255, 0, 0)

ContourOCX1.ResetVectorLinesColor

ContourOCX1.DrawVectorLines Picture1.hdc

.....





## 4 流线的部分屏蔽

```
(1) void InitializeForMaskVectorLines();
```

功能：屏蔽流线初始化

参数：无

```
(2) void MaskVectorLinesByFrame(long dc);
```

功能：部分屏蔽流线

参数：

dc: 设备环境句柄

```
(3) void DrawMaskedVectorLines(long dc);
```

功能：绘制经屏蔽后的流线

参数：

dc: 设备环境句柄

例如：

```
-----  
ContourOCX1.InitializeForMaskVectorLines  
Dim s As String  
s = "border"  
ContourOCX1.AddNewFrame s  
ContourOCX1.AddFrmCtrlPoint 130, 21  
ContourOCX1.AddFrmCtrlPoint 130, 35  
ContourOCX1.AddFrmCtrlPoint 121, 30  
Dim dc As Long  
dc = GetDC(Picture1.hwnd)  
ContourOCX1.MaskVectorLinesByFrame dc  
ContourOCX1.DrawMaskedVectorLines dc  
ReleaseDC Picture1.hwnd, dc  
-----
```

## 5 箭头矢量场绘制

```
(1) void SetArrowLengthRange(long minLength, long maxLength);
```

功能：设置箭头的长度范围

参数：minLength, maxLength: 最短长度和最长长度，以像素为单位

例如：

```
-----  
ContourOCX1.SetArrowLengthRange 1, 52
```

---

```
(2) void SetArrowSize(long size);
```

功能：箭头线的粗细

参数：size:粗细

```
(3) void SetArrowColor(long color);
```

功能：设置箭头的颜色

参数：color:颜色

```
(4) void DrawOneArrow(long hdc, float x, float y, long color, long size, long length);
```

功能：画一个箭头

参数：

dc:设备的 dc

x,y:位置

color:箭头的颜色

size:箭头线的粗细

length:箭头线的长度。如果是一1，则根据矢量模的大小取长短。

例如：

---

```
ContourOCX1.DrawOneArrow Picture1.hdc, j * 20, i * 20, RGB(128, 128, 128), 1, -1
```

---

```
(5) void DrawOriginalVectors(long hdc);
```

功能：画出原始输入点处的箭头

参数：

dc: 设备环境句柄

例如：

---

```
ContourOCX1.SetArrowColor RGB(255, 0, 0)
```

```
ContourOCX1.SetArrowLengthRange 1, 20
```

```
ContourOCX1.DrawOriginalVectors Picture1.hdc
```

---

## 八、内存的释放

```
void FreeData();
```

释放内存

## 九、 其它过程

### 1. 向屏幕输出图形时的重定位

```
(1) void SetDefaultPositionValus(long width, long height, float x1, float x2, float y1, float y2);
```

功能: 设置用户窗口与图形的坐标

参数:

width,height:用户窗口的宽和高

x1,x2,y1,y2: 要在:用户窗口中显示的图形的最小与最大坐标

```
(2) void GetScrPosFromRealPos(float RealX, float RealY, long* ScrX, long* ScrY);
```

功能: 由实际坐标获得屏幕上的坐标

参数:

RealX, RealY:实际坐标

ScrX, ScrY:屏幕坐标

```
(3) void GetRealPosFromScrPos(long ScrX, long ScrY, float* RealX, float* RealY);
```

功能: 由屏幕上的坐标获得实际坐标

参数:

ScrX, ScrY:屏幕坐标

RealX, RealY:实际坐标

### 2. 设置颜色级

```
(1) void ClearColorClass();
```

功能: 清除颜色级

参数: 无

```
(2) void SetDefaultZValueRange(float MinValue, float MaxValue);
```

功能: 设置 Z 值的变化范围

参数:

MinValue, MaxValue: Z 的最大与最小值

```
(3) void AddNewElementToColorClass(long ColorElement);
```

功能: 新加入一个颜色元素

参数:

ColorElement: 新颜色元素

```
(4) long GetValueColorInColorClass(float value);
```

功能 :得到一个颜色值

参数:

value:a 输入值

### 3 使等值面透明

```
void SetPolygonsTransparent(float transparent);
```

功能: 使等值面透明。

参数:

transparent:透明度系数

例如:

```
-----  
ContourOCX1.SetPolygonsTransparent 0.5
```

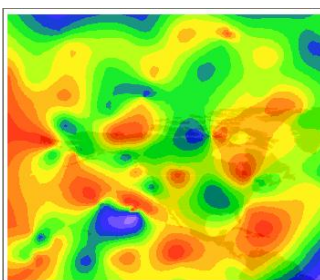
```
Dim dc As Long
```

```
dc = GetDC(Picture2.hwnd)
```

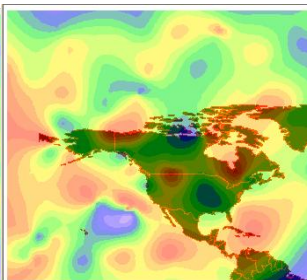
```
Picture2.Cls
```

```
ContourOCX1.DrawAllPolygons dc
```

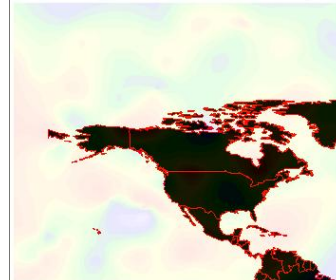
```
ReleaseDC Picture2.hwnd, dc  
-----
```



Transparent=0.1



Transparent=0.5



Transparent=0.9

### 4 输出插值后的数据

```
void SetOutputDirectionToDataFile( BSTR Path, float XStart, float XEnd, float XInterval, float  
YStart, float YEnd, float YInterval, long DataAndContour);
```

功能: 导出插值后的数据

Path: 数据文件的保存路径

XStart, XEnd, YStart, YEnd : 导出数据的范围

XInterval, Yinterval: 横向和纵向的步长

DataAndContour: 在 CalculateRandom 中是否执行等值线搜索

1—执行等值线搜索 0—不进行等值线搜索

例如:

```
ContourOCX1.SetOutputDirectionToDataFile "d:\xxx.xyz", 1, 100, 1, 1, 100, 1, 1
```

```
ContourOCX1.CalculateRandom
```

输出后的结果以二进制形式存贮。

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	
00000000h:	69	6E	74	65	72	70	00	00	64	00	00	00	00	00	00	00	interp..d...d...
00000010h:	00	00	80	3F	00	00	80	3F	00	00	80	3F	00	00	80	3F	..e?..e?..e?..e?
00000020h:	0D	01	00	00	DD	00	00	00	0C	01	00	00	DD	00	00	00	....?.....?..
00000030h:	0C	01	00	00	DC	00	00	00	0B	01	00	00	DC	00	00	00	....?.....?..
00000040h:	0A	01	00	00	DC	00	00	00	08	01	00	00	DB	00	00	00	....?.....?..
00000050h:	08	01	00	00	DB	00	00	00	08	01	00	00	DB	00	00	00	....?.....?..
00000060h:	08	01	00	00	DB	00	00	00	06	01	00	00	DA	00	00	00	....?.....?..
00000070h:	05	01	00	00	DA	00	00	00	05	01	00	00	D9	00	00	00	....?.....?..
00000080h:	04	01	00	00	D9	00	00	00	03	01	00	00	D8	00	00	00	....?.....?..
00000090h:	03	01	00	00	D8	00	00	00	02	01	00	00	D7	00	00	00	....?.....?..
000000a0h:	01	01	00	00	D7	00	00	00	01	01	00	00	D6	00	00	00	....?.....?..
000000b0h:	00	01	00	00	D5	00	00	00	FF	00	00	00	D5	00	00	00	....?.. ...?..
000000c0h:	FF	00	00	00	D4	00	00	00	FF	00	00	00	D3	00	00	00	...?.. ...?..
000000d0h:	FF	00	00	00	D3	00	00	00	FE	00	00	00	D2	00	00	00	...?..?..?..
000000e0h:	FE	00	00	00	D1	00	00	00	FE	00	00	00	D0	00	00	00	?..?..?..?..
000000f0h:	FE	00	00	00	D0	00	00	00	FE	00	00	00	CF	00	00	00	?..?..?..?..
00000100h:	EE	A6	E5	40	EE	A6	E5	40	EE	A6	E5	40	EE	A6	E5	40	香铈香铈香铈香铈
00000110h:	E6	9F	E5	40	42	95	E5	40	D6	86	E5	40	63	72	E5	40	镉铈B囑@葐铈cr铈
00000120h:	F0	55	E5	40	05	2F	E5	40	D3	7A	E4	40	65	B6	E4	40	镉铈./铈愈铈e朵@
00000130h:	9C	5E	E4	40	4D	F0	E3	40	2F	68	E3	40	19	C3	E2	40	秉铈M塍@/h站.免@

安装 ContourOCX5.23 版本以上后，系统中会自动加入一个新的类库（InterpolateDataBlock），用与读取此二进制文件。

```
Dim f As New InterpolateDataBlock
f.Read "d:\xxx.xyz"
Dim row As Long, col As Long
Dim x As Single, y As Single, z As Single
f.GetRowCol row, col
For i = 0 To row - 1
  For j = 0 To col - 1
    f.GetPoint i, j, x, y, z
    Debug.Print x, y, z
  Next
Next
```

## 5 使用 DELPHI 中出现异常的特别处理

如果您在 Delphi 7 以下的版本中使用该控件，在没有安装必要的升级包的情况下，有可能会出现以下错误提示：



所以有必要采取以下措施：

1. 声明全局的变量保存原有的 8087CW 设置

```
var
  Saved8087CW: Word;
```

2. 在 mainForm 的 OnCreate 过程中改变 8087CW 的值, 改变之前先保存默认值

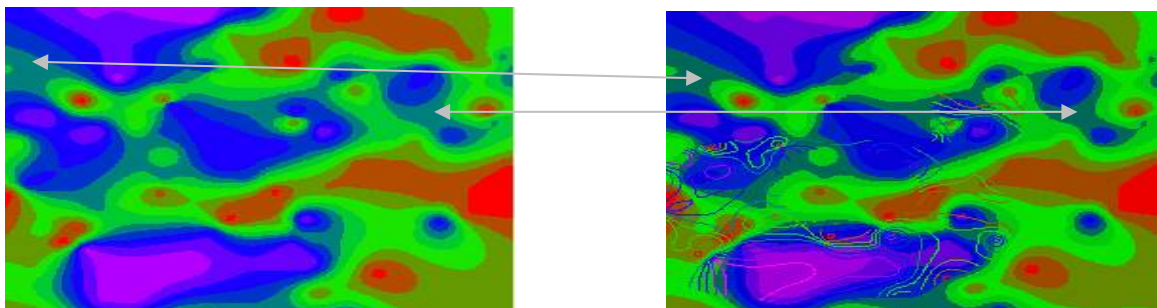
```
Saved8087CW := Default8087CW;
Set8087CW($133f);
```

3. 在程序结束前, 使用默认值重置 8087CW 指令值。在 mainForm 的 OnClose 过程中

```
Set8087CW(Saved8087CW);
```

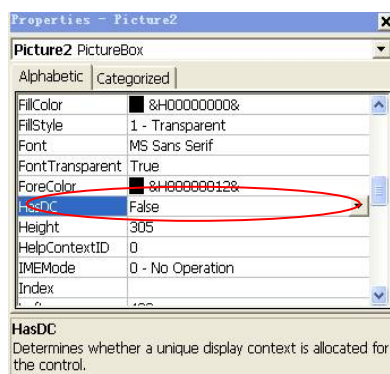
## 6 VB 中 MASK 后颜色改变的问题

在对等值线或等值面进行 MASK 操作后, 图上的某些颜色会发生改变。如下:



请采取以下措施:

- (1) 请将 PictureBox 的 HasDC 设为 false,



- (2) 在绘图函数的使用参数要用 API 函数取得。

```
如: 'draw  
Dim dc As Long  
dc = GetDC(Picture2.hwnd)  
ContourOCX1.MaskLinesByFrame dc  
ContourOCX1.DrawMaskedLines dc  
ReleaseDC Picture2.hwnd, dc
```

## 十、联系方式

如感兴趣或有任何问题和建议或有意购买，请与以下地址联系：  
[cui1012@sh163.net](mailto:cui1012@sh163.net)或[cuixuesen@eastfishery.ac.cn](mailto:cuixuesen@eastfishery.ac.cn) .谢谢。  
网站: <http://snowedforest.brinkster.net/>