

VISUAL BASIC CLASS FOR SIMPLE SURFACE REPRESENTATION WITH HIDDEN FACE REMOVAL

ANTAL Tiberiu Alexandru

Abstract. The paper presents a set of formulae for simple surfaces representation with hidden face removal implemented in Visual BASIC with the help of object-oriented methodologies.

1. OBLIQUE PROJECTIONS

Generally speaking, a projection transforms points in an N system to points in a $(N-1)$ system. Most of the practical problems are focusing on mapping a 3D object to a 2D display surface, also called projection plane (PP). Projection rays (projectors) emanate from a Center of Projection (COP) and intersect Projection Plane (PP). The COP for parallel projectors is at infinity. In the oblique projection (*Figure 1*) the projectors are not perpendicular to the projection plane but are parallel from the object to the Projection Plane.

The projectors are defined by two angles α and θ where:

- α - angle of line (x, y, x_p, y_p) with the projection plane (PP),
- θ - angle of line (x, y, x_p, y_p) with x axis in the projection plane,
- L - Length of Line (x, y, x_p, y_p) .

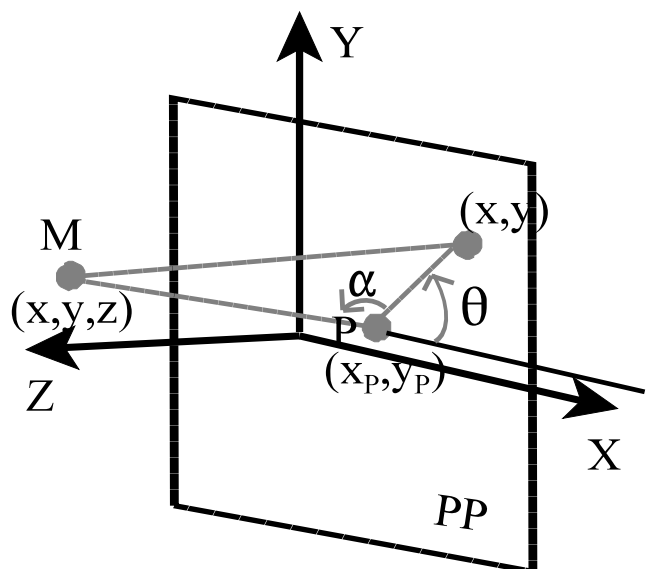


Figure 1 - The oblique projection of point (x, y, z) .

Then:

$$\begin{aligned}\cos(\theta) &= \frac{(x-x_p)}{L}, \\ \sin(\theta) &= \frac{(y-y_p)}{L}, \\ \tan(\alpha) &= \frac{z}{L}.\end{aligned}\tag{1}$$

Defining $S = \frac{L}{z}$ we have:

$$\begin{aligned}x_p &= x - z \cdot S \cdot \cos(\theta) \\ y_p &= y - z \cdot S \cdot \sin(\theta)\end{aligned}\tag{2}$$

2. SIMPLE SURFACES

In mathematics a surface is a function that returns a Z value for each X and Y coordinates in region of interest. The class defined in Visual BASIC considers the surface defined by a function that return a Y value for each X and Z coordinate in a region. To translate a function from a system that gives Z as a function of X and Y the roles of Y and Z will be reversed, then the (2) formulae become:

$$\begin{aligned}x_p &= x - y \cdot S \cdot \cos(\theta) \\ y_p &= z - y \cdot S \cdot \sin(\theta)\end{aligned}\tag{3}$$

For each X and Z value, a simple surface can have at most one Y value. Simple objects are such as planes, pyramids, and bowls. Objects that are not included in the "simple surface" category are like spheres, cubes, and tetrahedrons as they have more than one Y value corresponding to the same X and Z coordinates.

3. OBJECT-ORIENTED PROGRAMMING WITH Visual BASIC 6

3.1 General issues of the Object-oriented methodologies

Object-oriented (OO) methodologies allow us to model the real things as objects. Object modeling consists of looking for objects. Objects can be described by their attributes and operations. Attributes are changeable characteristics, while operations are the things an object does or can have done to it. The four elements of an OO system are abstraction, encapsulation, inheritance, and polymorphism. We use abstraction to identify the objects involved in a particular application. This way we can focus on an application's objects rather than its implementation. Encapsulation aids in

abstraction by hiding the internal implementation of the object within the class. An object can then be used without understanding of how the object's class is implemented. The object can be changed only through one of its operations. Operations and attributes are encapsulated in a single definition. As access to data is controlled, inadvertent or illegal transformation can't be made. Classes can be arranged in inheritance hierarchies. A class inherits operations and attributes from its parent. This is one of the underlying mechanisms for providing code reusability. Polymorphism allows that operation with the same name may be defined in many places. The only operations of an object are those which are defined by the object or inherited from a parent or generally speaking from all ancestors.

3.2 Classes in Visual BASIC

In Visual BASIC attributes are called properties and are defined with variables and property procedures. Operations, also called methods in Visual BASIC, that define the behavior of the class are implemented using function and sub procedures. Property procedures provide the public interface to the class's private properties. With property procedures attribute values can be set or get, or set a reference to an object.

The class `Name` property must be set to `Class3DSurf` in order to use it in any Visual BASIC application and has the following code:

```
Option Explicit
Private Type punct3D
    X As Single
    Y As Single
    Z As Single
End Type
Private Type POINTAPI
    X As Long
    Y As Long
End Type

Private mXmin As Single
Private mYmin As Single
Private mXmax As Single
Private mYmax As Single
Private mnx As Integer
Private mny As Integer
Private pts(1 To 4) As POINTAPI
Private mpuncte3d() As punct3D

Public s As Single
Public th As Single
Public Ascunde As Boolean

Private Declare Function Polygon Lib "gdi32" (ByVal hdc As Long, _
    lpPoint As POINTAPI, ByVal nCount As Long) As Long

'used to create the array that holds the surface points
Public Sub SetareValori(ByVal x1 As Single, ByVal x2 As Single, _
```

```
pasx As Single, ByVal y1 As Single, ByVal y2 As Single, _
ByVal pasy As Single, ByRef zxy() As Single)
```

```
Dim X As Single
Dim Y As Single
Dim i As Integer
Dim j As Integer
Dim c As Long
mnx = (x2 - x1) / pasx
mny = (y2 - y1) / pasy
ReDim mpuncte3d(mnx, mny)
c = 0
For i = 0 To mnx
    For j = 0 To mny
        mpuncte3d(I, j).X = zxy(c)
        mpuncte3d(I, j).Y = zxy(c + 1)
        mpuncte3d(I, j).Z = zxy(c + 2)
        c = c + 3
    Next j
Next i
```

```
End Sub
```

```
Private Sub Transformare3d2D()
```

```
Dim i As Integer
```

```
Dim j As Integer
```

```
For i = 0 To mnx
    For j = 0 To mny
        mpuncte3d(I, j).X = mpuncte3d(I, j).X - _
            mpuncte3d(I, j).Y * Cos(th) * s
        mpuncte3d(I, j).Y = mpuncte3d(I, j).Z - _
            mpuncte3d(I, j).Y * Sin(th) * s
        mpuncte3d(I, j).Z = 0
    Next j
Next i
```

```
End Sub
```

```
Private Sub Minmax()
```

```
Dim i As Integer
```

```
Dim j As Integer
```

```
mXmin = mpuncte3d(0, 0).X
```

```
mXmax = mpuncte3d(0, 0).X
```

```
mYmin = mpuncte3d(0, 0).Y
```

```
mYmax = mpuncte3d(0, 0).Y
```

```
For i = 0 To mnx
```

```
    For j = 0 To mny
```

```
        If mpuncte3d(I, j).X > mXmax Then mXmax = _
```

```

        mpuncte3d(I, j).X
    If mpuncte3d(I, j).X < mXmin Then mXmin = _
        mpuncte3d(I, j).X
    If mpuncte3d(I, j).Y > mYmax Then mYmax = _
        mpuncte3d(I, j).Y
    If mpuncte3d(I, j).Y < mYmin Then mYmin = _
        mpuncte3d(I, j).Y
    Next j
Next i
Debug.Print mXmin, mXmax, mYmin, mYmax
End Sub

Private Sub scalare2D(pic1 As PictureBox)
    Dim lx As Single
    Dim ly As Single
    Dim i As Integer
    Dim j As Integer
    ly = pic1.ScaleHeight
    lx = pic1.ScaleWidth
    For i = 0 To mnx
        For j = 0 To mny
            'points of a face
            mpuncte3d(I, j).X = scalare(mpuncte3d(I, j).X, _
                mXmin, mXmax) * lx
            mpuncte3d(I, j).Y = ly - scalare(mpuncte3d(I, j).Y, _
                mYmin, mYmax) * ly
        Next j
    Next i
End Sub

Private Function scalare(X As Single, mXmin As Single, _
    mXmax As Single) As Single
    Dim d As Single
    d = mXmax - mXmin
    If d = 0 Then d = 1
    scalare = (X - mXmin) / d
End Function

Public Sub Grafic(ByVal pic1 As PictureBox)
    Dim i As Integer
    Dim j As Integer
    Call Transformare3d2D
    Call Minmax
    Call scalare2D(pic1)
    If Ascunde Then
        pic1.FillStyle = vbFSSolid
        pic1.FillColor = vbWhite
    Else
        pic1.FillStyle = vbFSTransparent
    End If
End Sub

```

```

For i = 0 To mnx - 1
  For j = 0 To mny - 1
    'points of a face
    pts(1).X = mpuncte3d(I, j).X
    pts(1).Y = mpuncte3d(I, j).Y
    pts(2).X = mpuncte3d(I + 1, j).X
    pts(2).Y = mpuncte3d(I + 1, j).Y
    pts(3).X = mpuncte3d(I + 1, j + 1).X
    pts(3).Y = mpuncte3d(I + 1, j + 1).Y
    pts(4).X = mpuncte3d(I, j + 1).X
    pts(4).Y = mpuncte3d(I, j + 1).Y
    Polygon pic1.hdc, pts(1), 4
  Next j
Next i

```

End Sub

The control names from *Figure 2* are working together with the following code and the class from above to draw a surface.

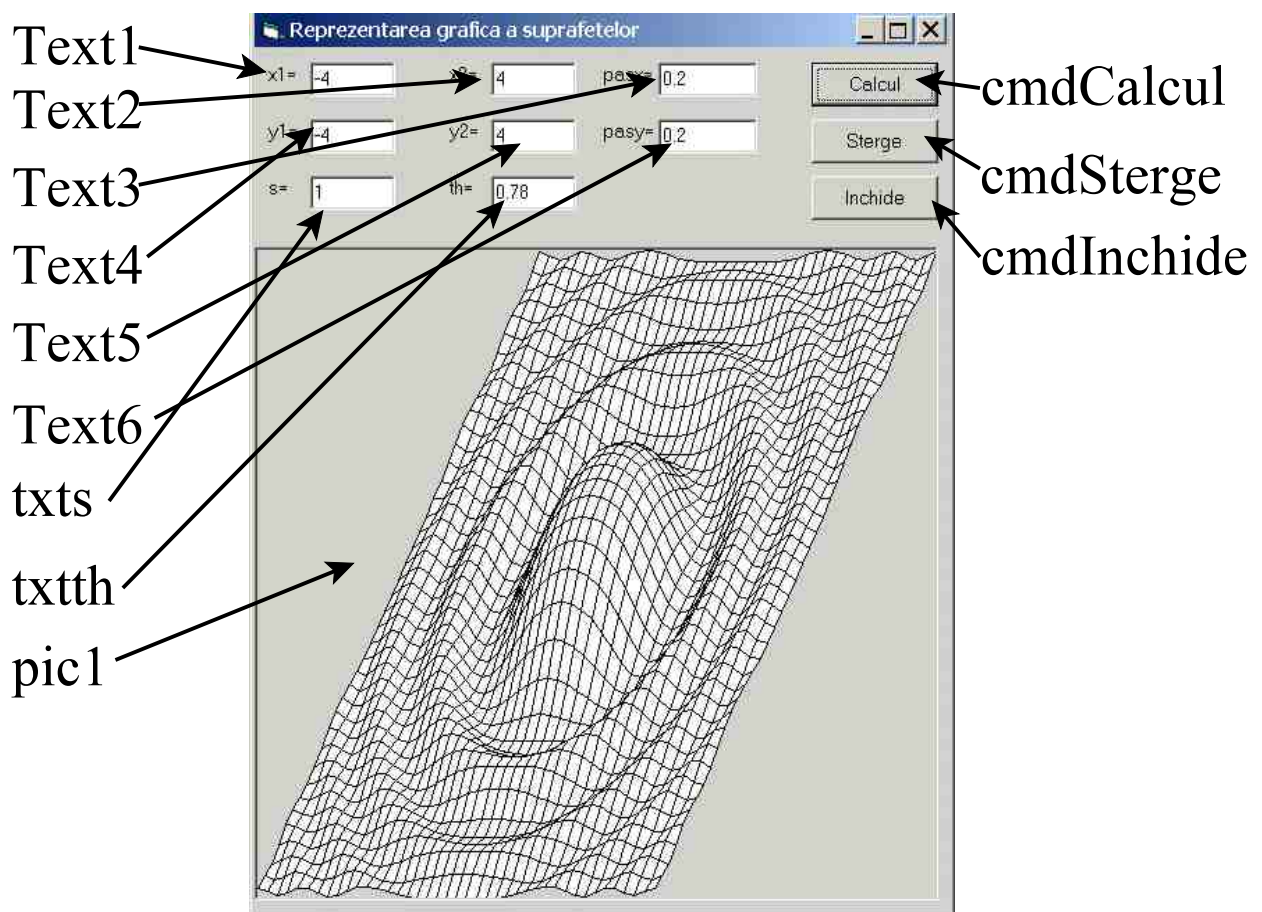


Figure 2 - The control names used for the data input in the surface representation from.

The code corresponding to the **Click** event for the **CommandButton** control **Calcul** from *Figure 2* is the following:

```

Private Sub cmdCalcul_Click()
    Dim gr3d As Class3DSurf
    Dim nx As Integer
    Dim ny As Integer
    Dim i As Integer
    Dim j As Integer
    Dim x1 As Single
    Dim x2 As Single
    Dim X As Single
    Dim pasx As Single
    Dim y1 As Single
    Dim y2 As Single
    Dim Y As Single
    Dim pasy As Single
    Dim c As Long
    Dim zxy() As Single

    x1 = CSng(Text1.Text)
    x2 = CSng(Text2.Text)
    pasx = CSng(Text3.Text)
    nx = (x2 - x1) / pasx
    y1 = CSng(Text4.Text)
    y2 = CSng(Text5.Text)
    pasy = CSng(Text6.Text)
    ny = (y2 - y1) / pasy

    ReDim zxy(3 * (nx + 1) * (ny + 1))
    X = x1
    c = 0
    For i = 0 To nx
        Y = y1
        For j = 0 To ny
            zxy(c) = X
            zxy(c + 1) = Y
            zxy(c + 2) = f(X, Y)
            Y = Y + pasy
            c = c + 3
        Next j
        X = X + pasx
    Next i
    'the class instantiation
    Set gr3d = New Class3DSurf
    gr3d.SetareValori x1, x2, pasx, y1, y2, pasy, zxy
    gr3d.s = CSng(txts)
    gr3d.th = CSng(txtth)
    gr3d.Ascunde = True
    gr3d.Grafic frmSuprafata.pic1
End Sub

Private Sub cmdInchide_Click()

```

```

Unload Me
End Sub

Private Function f(X As Single, Y As Single) As Single
    'equation of the surface
    f = Sin(1 + X * X + Y * Y) / (1 + X * X + Y * Y)
End Function

Private Function f1(X As Single, Y As Single) As Single
    'intersoction of two surfaces
    Dim Z As Single
    Z = 1 - Y
    If Y < X Then Z = 1 - X
    f1 = Z
End Function

Private Function f2(X As Single, Y As Single) As Single
    'intersection of four surfaces
    Dim Z As Single
    Dim r As Single
    Z = 1 - Y
    If Y < X Then Z = 1 - X
    r = Sqr((1.5 - X) ^ 2 + (1.5 - Y) ^ 2)
    If r < 1.5 Then Z = 0
    f5 = Z
End Function

Private Sub cmdSterge_Click()
    frmSuprafata.pic1.Cls
End Sub

```

4. RESULTS

Representations of some simple surfaces are presented in the following figures. Function **f** was used to obtain the representations from *Figure 3*, *Figure 4*, *Figure 5*, and *Figure 6*. Here the s and θ (th) parameters of the oblique projection were changed to show their effects on the projection.

Figure 7 was obtained using function **f1**, and *Figure 8* was obtained using function **f2**.

In order to remove the hidden part of the surfaces I used the **Polygon** API function. This draws a polygon consisting of an arbitrary series two or more points. Windows will automatically close the polygon by connecting the last and first points. The point must be provided according to the POINTAPI Windows API data type. The border of the polygon is drawn with the currently-selected pen, and filling is made using the currently-selected brush. In order to make the surface more realistic and easier to understand faces that are the furthest from the center of the projection are drawn first, while those that are closer are left at the end. As the closer faces are the last drawn, they will cover some parts of those that are already drawn.

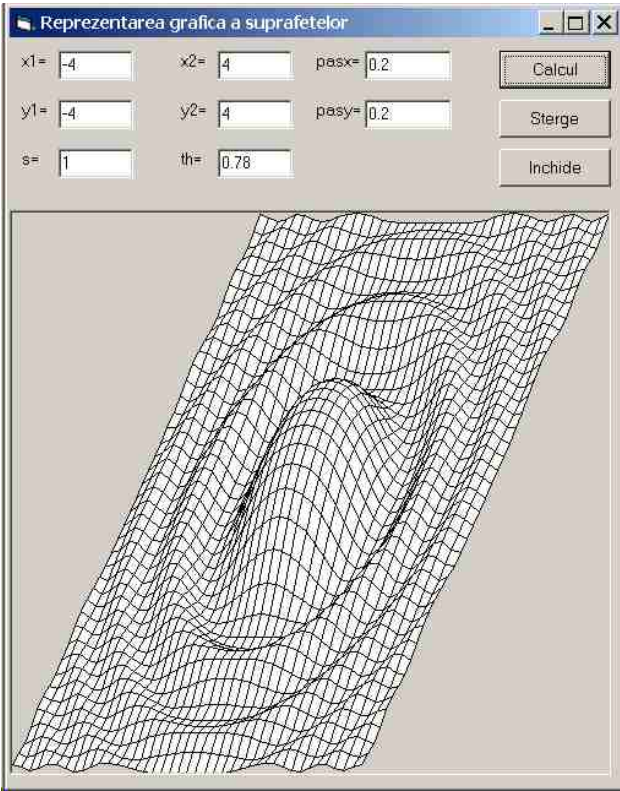


Figure 3

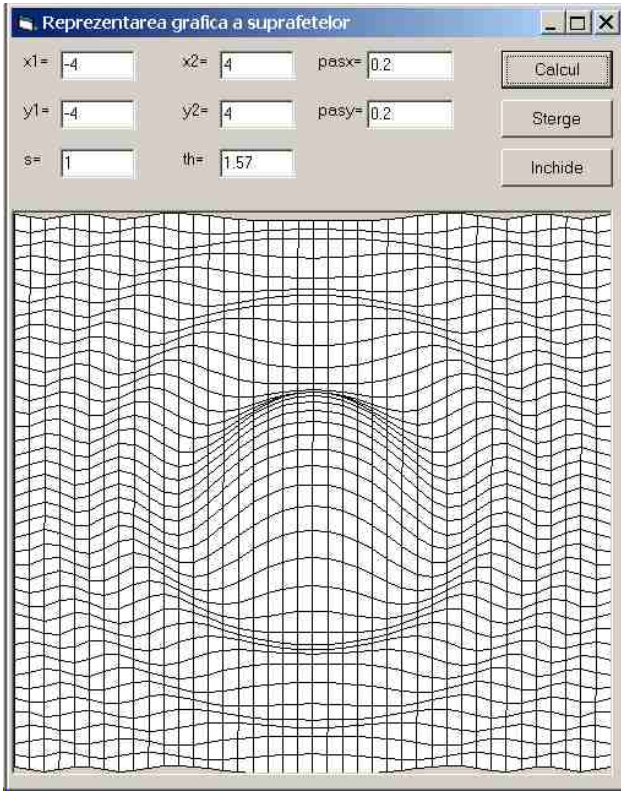


Figure 4

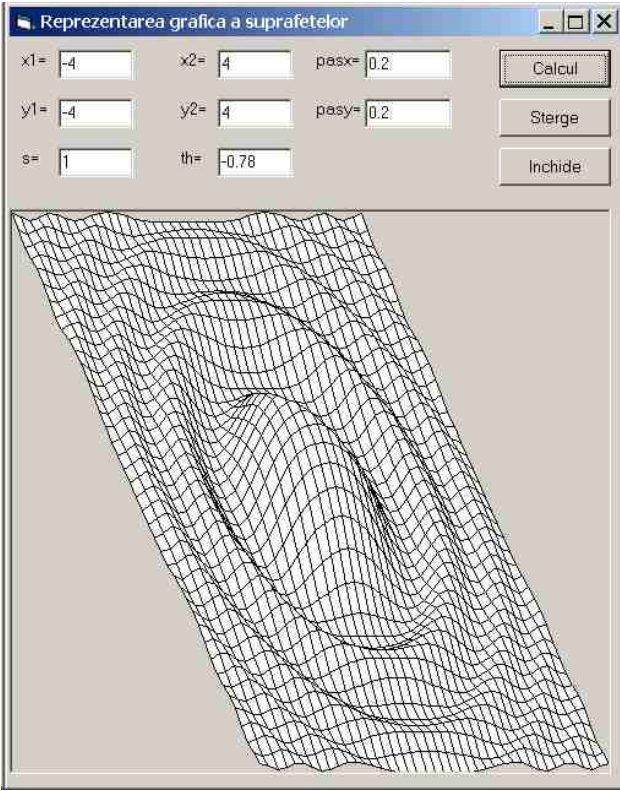


Figure 5

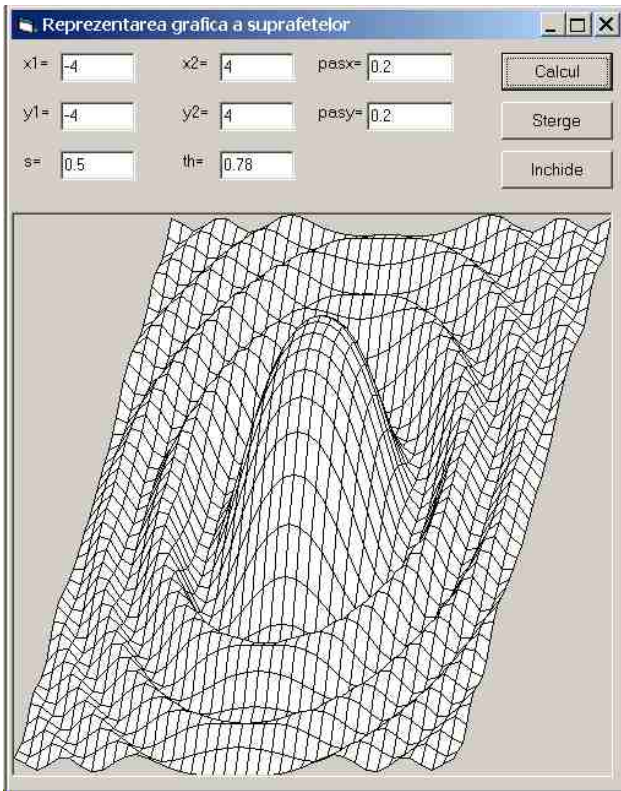


Figure 6

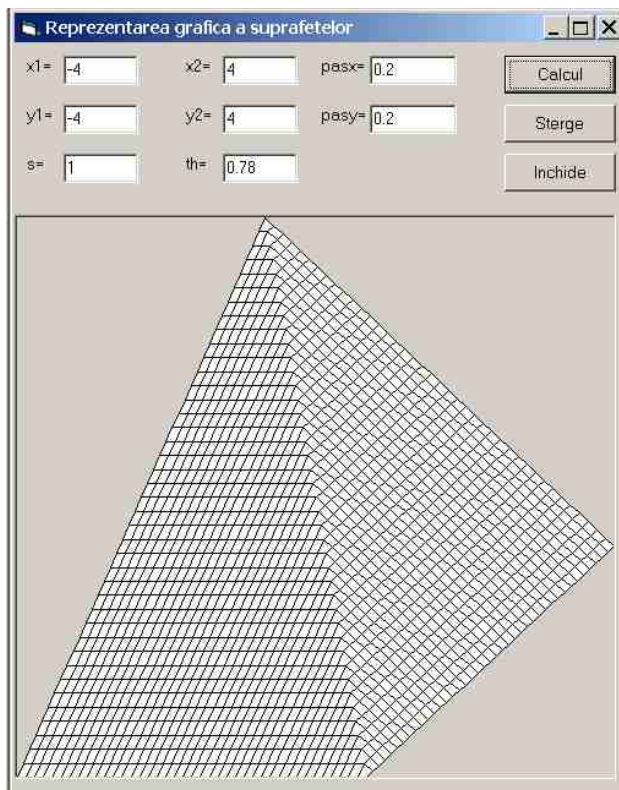


Figure 7

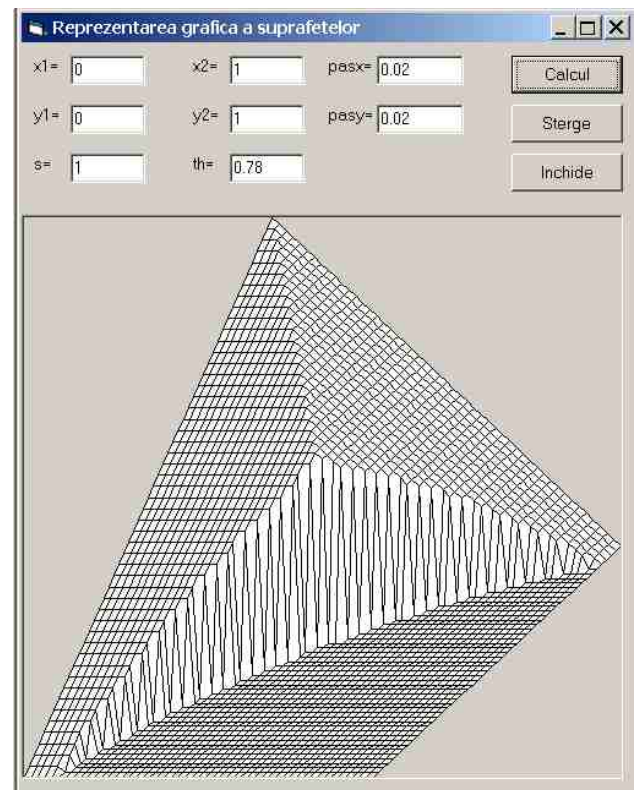


Figure 8

REFERENCES

- [1] Appleman, D., Visual Basic Programmer's Guide to the Win32 API, SAMS, 1999, ISBN 0-672-31590-4.
- [2] Getz, K., Gilbert, M., VBA Developer's Handbook, Sybex, 2000, ISBN: 0-7821-2978-1.
- [3] Reverchon, A., Ducamp, M., Mathematique sur Micro-ordinateur, EYROLLES, 1986.
- [4] McFedries, P., Visual Basic for Applications, SAMS, 1997 ISBN: 0-672-31046-5.

Clasă implementată în limbajul Visual BASIC pentru reprezentarea grafică a suprafețelor simple cu ascunderea fețelor invizibile

Lucrarea prezintă implementarea unei clase în limbajul Visual BASIC în vederea reprezentării grafice a unor suprafețe simple cu ascunderea fețelor invizibile. Desenarea suprafeței se face prin fețe, iar ascunderea rezultă din ordinea afișării acestora.

Senior lecturer dr. eng. ANTAL Tiberiu Alexandru, Technical University of Cluj-Napoca, Department of Mechanics and Computer Programming, Faculty of Machine Building, B-dul Muncii, Nr. 103-15, Cluj-Napoca, RO-3400, ROMANIA.