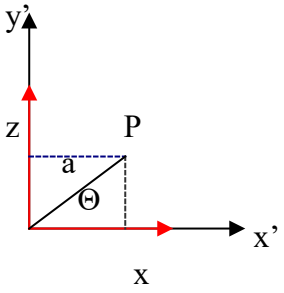
# 3D REPRESENTATION ON PLAIN SURFACE


***BY***

***"A Strange Site"***
***http://astrangesite.altervista.org***

***powered by***
***dr. Alessandro Strano - Italy***


We can represent the generic point P(z, x, y) of a three dimension space in a two dimension space transforming the three coordinates in a couple of values x' and y' using "orthogonal projections".




Imagine you are on y[1] axis and look at point P(z, x, y); the graphic above shows this point of view. Well, we can calculate our values by means of trigonometry.

We know that:
$y' = a * \text{sen}(\Theta + \beta)$
$x' = a * \cos(\Theta + \beta)$

[1] where x, y and z are axes of 3D orthogonal system; while x' and y' are axes of 2D orthogonal system.

Where Θ is the angle formed by segment "a" and x axis, while β is the rotation angle of orthogonal system round y axis (note that in our example β is zero).

Applying formulas of goniometric addition[2] we obtain:
$y' = a^* \operatorname{sen} \Theta * \cos \beta + a * \cos \Theta * \operatorname{sen} \beta = z * \cos \beta + x * \operatorname{sen} \beta$
$x' = a * \cos \Theta * \cos \beta - a * \operatorname{sen} \Theta * \operatorname{sen} \beta = x * \cos \beta - z * \operatorname{sen} \beta$

At last, if we turn the orthogonal system round z axis of an angle α we have to (applying the same procedure) replace in our formulas the "x" with the following expression: $x * \cos \alpha - y * \operatorname{sen} \alpha$.

These are our formulas:
$k = x * \cos \alpha - y \operatorname{sen} \alpha$
$y' = z * \cos \beta + k * \operatorname{sen} \beta$
$x' = k * \cos \beta - z * \operatorname{sen} \beta$

where x' e y' are the coordinates we was looking for.

The following Visual Basic[3] code is an easy sample programme to draw 3d functions: it is based on formulas given above. If you want to check it, simple make a "form" with a "command" control in which "Click()" trigger you have to copy the code. I chose to draw a segment (parallel to y' axis) among points except, of course, after a discontinuity point; the "flag" TracciaLinea and the error handling procedure make just that. If you want to draw graphics of square functions[4] you have to execute the procedure both for the point P(z ,x , y) and the point P(-z, x, y), so if you want draw segments, parallel to x' axis, among points, you have to add, after "next x", an other cycle similar to the first, but "for y = .." will be placed before "for x = .." and the error handling procedure must be changed according to that. Note that the method I suggested is only one of the many possible

[2] $\operatorname{sen}(\Theta + \beta) = \operatorname{sen} \Theta * \cos \beta + \cos \Theta * \operatorname{sen} \beta$; $\cos(\Theta + \beta) = \cos \Theta * \cos \beta - \operatorname{sen} \Theta * \operatorname{sen} \beta$.

Cfr. G. Zwirner – L. Scaglianti, Matematica per ragionieri periti commerciali e programmatori, Padova 1984, vol. II, p. 46.

[3] "Visual Basic" is a trademark of Microsoft Corporation.

[4] For example the function z^2 = x^2 + y^2 that we will write as z = sqr(x^2 + y^2).

solutions, perhaps the most intuitive. A different approuch is, for example, explained on C. Cremonesi (Algoritmi di calcolo numerico, Milano 1990, pp. 6-12) where hidden part of graphics are not drawn.

```
On Error GoTo GestErrore
Dim x As Double, y As Double, z As Double
Dim x1 As Double, y1 As Double, k As Double, a As Double, b As Double
Dim l As Double, w As Double, r As Double, PiGreco As Double
Dim TracciaLinea As Boolean, zoom As Double
Cls
PiGreco = 4 * Atn(1)
w = Width / 2
l = Height / 2
'--------------------
' a= rotation angle round z axis (degrees)
' b= rotation angle round y axis (degrees)
' zoom = zoom
' These values and those of the two
' cycles FOR could be passed as parameters
'--------------------
a = 25
b = 0
zoom = w / 5
'--------------------
a = a * PiGreco / 180  'angle in radians
b = b * PiGreco / 180 'angle  in radians
TracciaLinea = False
For x = -2.5 To 2.5 Step 0.1
   For y = -2.5 To 2.5 Step 0.1

      '------------------------------
      'write here the function
      ‘ you like
      '------------------------------
      z = 3 * Sin(x) ^ 3 * Sin(y) ^ 3
      '------------------------------

      k = x * Cos(a) - y * Sin(a)
      y1 = z * Cos(b) + k * Sin(b)
      x1 = k * Cos(b) - z * Sin(b)
      x1 = (w + x1 * zoom)
      y1 = (l - y1 * zoom)
```

```
            If TracciaLinea Then
               Line -(x1, y1)
            Else
               PSet (x1, y1)
            End If
            TracciaLinea = True
ProssimoValore:
         Next y
      Next x
      Exit Sub
GestErrore:
      TracciaLinea = False
      Resume ProssimoValore
```