Subject: Re: Displaying Cartesian coordinate data on a sphere
Posted by dplatten on Wed, 12 Dec 2012 09:53:28 GMT
View Forum Message <> Reply to Message

I have found the following solution to be more reliable. It uses the /GRID, XOUT and YOUT switches to ensure that the resulting grid covers the whole sphere. I've realised that the coordinates of the resulting grid are in terms of longitude and latitude. To ensure that I finished up with a grid that covers the entire surface of a sphere I set XOUT to cover a range of +/- PI radians, and YOUT to cover +/- PI/2 radians.

```
x = [0,  0,  1, -1,  0,  0]
y = [0,  0,  0,  0,  1, -1]
z = [1, -1,  0,  0,  0,  0]
values = [255, 0, 100, 50, 0, 150]
maxOut = 2.0*!pi
xout = FINDGEN(31) * 2.0*!pi / 30.0 - (2.0*!pi / 2.0)
yout = FINDGEN(31) * !pi / 30.0 - (!pi / 2.0)
grid = GRIDDATA(x, y, z, values, /GRID, XOUT=xout, YOUT=yout, /SPHERE)
cgLoadCT, 33
cgWindow, 'cgImage', grid, /KEEP_ASPECT

image = BYTSCL(grid)
MESH_OBJ, 4, vertices, polygons, REPLICATE(0.25, 101, 101)
oModel = OBJ_NEW('IDLgrModel')
oPalette = OBJ_NEW('IDLgrPalette')
oPalette -> LOADCT, 33
oPalette -> SetRGB, 255, 255, 255, 255
oImage = OBJ_NEW('IDLgrImage', image, PALETTE = oPalette)
vector = FINDGEN(101)/100.
texure_coordinates = FLTARR(2, 101, 101)
texure_coordinates[0, *, *] = vector # REPLICATE(1., 101)
texure_coordinates[1, *, *] = REPLICATE(1., 101) # vector
oPolygons = OBJ_NEW('IDLgrPolygon',  $
   DATA = vertices, POLYGONS = polygons, $
   COLOR = [255, 255, 255], $
   TEXTURE_COORD = texure_coordinates, $
   TEXTURE_MAP = oImage, /TEXTURE_INTERP)
oModel -> ADD, oPolygons
oModel -> ROTATE, [1, 0, 0], -90
oModel -> ROTATE, [0, 1, 0], -90
XOBJVIEW, oModel
```

Regards,

David