```
''version 13-may-2010 [ not jet errorles ]

dim sk( 200 , 2 ) , pnt( 255 , 3 )
global bmx,bmy,bmz,bdx,bdy,bdz,frame
global black , red , green , yellow
global blue , magenta , cyan , white
global pink , purple , gray , orange
black = rgb( 0 , 0 , 0 )
red = rgb( 255 , 0 , 0 )
green = rgb( 0 , 255 , 0 )
yellow = rgb( 255 , 255 , 0 )
blue = rgb( 0 , 0 , 255 )
magenta = rgb( 255 , 0 , 255 )
cyan = rgb( 0 , 255 , 255 )
white = rgb( 255 , 255 , 255 )
pink = rgb( 255 , 127 , 127 )
orange = rgb( 255 , 127 , 0 )
gray = rgb( 127 , 127 , 127 )
purple = rgb( 127 , 0 , 127 )
pi = atn( 1 ) * 4
GOSUB [Initialize]' Do not remove this line
'========================== put code below





'========================== put code above
end
''----------------begin blua 3d
sub box mx,my,mz,dx,dy,dz
''fil center and border of any shape
bmx = mx
bmy = my
bmz = mz
bdx = dx
bdy = dy
bdz = dz
end sub
sub cilinder sides , dx , dz , kl1 , kl2
''create a cilider - shape
''dx dz = d top
''if kl1 if sides if sides > 32 then sides = 32
''first fil the swarm
for i = 0 to sides
call point i _
```

```
, bmx+sin(i*pi*2/sides)*bdx _
, bmy-bdy _
, bmz+cos(i*pi*2/sides)*bdz _
, rainbow(i*360/sides)
call point i + sides + 1 _
, bmx+sin(i*pi*2/sides)*dx _
, bmy+bdy _
, bmz+cos(i*pi*2/sides)*dz _
, rainbow(i*360/sides)
next i
''then create the sides
if kl1 for i = 0 to sides
call h4kl4 i,i+1,i+sides+2,i+sides+1
next i
else
for i = 0 to sides step 2
call h4kl1 i,i+1,i+sides+2,i+sides+1,kl1
call h4kl1 i+1,i+2,i+sides+3,i+sides+2,kl2
next i
end if
end sub
sub cube l , d , f , r , u , b
''create a cube-shape
''first fil the swarm
call point 0,bmx-bdx,bmy-bdy,bmz-bdz,0
call point 1,bmx-bdx,bmy-bdy,bmz+bdz,0
call point 2,bmx-bdx,bmy+bdy,bmz-bdz,0
call point 3,bmx-bdx,bmy+bdy,bmz+bdz,0
call point 4,bmx+bdx,bmy-bdy,bmz-bdz,0
call point 5,bmx+bdx,bmy-bdy,bmz+bdz,0
call point 6,bmx+bdx,bmy+bdy,bmz-bdz,0
call point 7,bmx+bdx,bmy+bdy,bmz+bdz,0
''then create the sides
call h4kl1 0,1,3,2,l''left
call h4kl1 7,6,4,5,r''right
call h4kl1 0,2,6,4,f''front
call h4kl1 7,5,1,3,b''back
call h4kl1 0,1,4,5,d''down
call h4kl1 7,6,3,2,u''up
end sub
sub setcolor kl
r = kl and 255
g = int( kl / 256 ) and 255
b = int( kl / 256 ^ 2 ) and 255
CALL glColor4fv r/256 , g/256 , b/256 , 1
end sub
```

```
sub point no , x , y , z , kl
''set a point in the swarm
if no  255 then exit sub
pnt( no , 0 ) = x
pnt( no , 1 ) = y
pnt( no , 2 ) = z
pnt( no , 3 ) = kl
end sub
sub h3kl1 p1 , p2 , p3 , kl
''create a 1 color triangle form swarm
call glBegin GL.TRIANGLES
call setcolor kl
CALL glVertex pnt(p1,0),pnt(p1,1),pnt(p1,2)
CALL glVertex pnt(p2,0),pnt(p2,1),pnt(p2,2)
CALL glVertex pnt(p3,0),pnt(p3,1),pnt(p3,2)
call glEnd
end sub
sub h3kl3 p1 , p2 , p3
''create a 3 color triangle from swarm
call glBegin GL.TRIANGLES
call setcolor pnt(p1,3)
CALL glVertex pnt(p1,0),pnt(p1,1),pnt(p1,2)
call setcolor pnt(p2,3)
CALL glVertex pnt(p2,0),pnt(p2,1),pnt(p2,2)
call setcolor pnt(p3,3)
CALL glVertex pnt(p3,0),pnt(p3,1),pnt(p3,2)
call glEnd
end sub
sub h4kl1 p1 , p2 , p3 , p4 , kl
''create a 1 color quadangle from swarm
call glBegin GL.QUADS
call setcolor kl
CALL glVertex pnt(p1,0),pnt(p1,1),pnt(p1,2)
CALL glVertex pnt(p2,0),pnt(p2,1),pnt(p2,2)
CALL glVertex pnt(p3,0),pnt(p3,1),pnt(p3,2)
CALL glVertex pnt(p4,0),pnt(p4,1),pnt(p4,2)
call glEnd
end sub
sub h4kl4 p1 , p2 , p3 , p4
''create 4 color quadangle from swarm
call glBegin G.QUADS
call setcolor pnt(p1,3)
CALL glVertex pnt(p1,0),pnt(p1,1),pnt(p1,2)
call setcolor pnt(p1,3)
CALL glVertex pnt(p2,0),pnt(p2,1),pnt(p2,2)
call setcolor pnt(p1,3)
```

```
CALL glVertex pnt(p3,0),pnt(p3,1),pnt(p3,2)
call setcolor pnt(p1,3)
CALL glVertex pnt(p4,0),pnt(p4,1),pnt(p4,2)
call glEnd
end sub
function rad( deg )
''get radians from degrees
rad = deg * pi / 180
end function
function rainbow( deg )
''ful circle = rainbow efect
rainbow = rgb( sin( rad( deg ) ) * 127 + 128 _
, sin( rad( deg - 120 ) ) * 127 + 128 _
, sin( rad( deg + 120 ) ) * 127 + 128 )
end function
function rgb( r , g , b )
''create a color
''r g b = 0...255
rgb = ( r and 255 ) _
+ ( g and 255 ) * 256 _
+ ( b and 255 ) * 256 * 256
end function
function mix( kl1 , f , kl2 )
''get a color between 2 colors
''kl1 kl2 = colors
''f = 0...1
r1 = int( kl1 and 255 )
g1 = int( kl1 / 256 ) and 255
b1 = int( kl1 / 256 / 256 ) and 255
r2 = int( kl2 and 255 )
g2 = int( kl2 / 256 ) and 255
b2 = int( kl2 / 256 / 256 ) and 255
dr = r2 - r1
dg = g2 - g1
db = b2 - b1
dr = dr * f
dg = dg * f
db = db * f
r = r1 + dr
g = g1 + dg
b = b1 + db
mix = rgb( r and 255 , g and 255 , b and 255 )
end function
''----------------end blua 3d
[Initialize]
NOMAINWIN
```

```
GLOBAL GL.POINTS : GL.POINTS = 0
GLOBAL GL.LINES : GL.LINES = 1
GLOBAL GL.TRIANGLES : GL.TRIANGLES = 4
GLOBAL GL.TRIANGLE.STRIP : GL.TRIANGLE.STRIP = 5
GLOBAL GL.QUADS : GL.QUADS = 7
GLOBAL GL.LINE.STRIP : GL.LINE.STRIP = 3
GLOBAL GL.TRIANGLE.STRIP : GL.TRIANGLE.STRIP = 5
GLOBAL GL.QUAD.STRIP : GL.QUAD.STRIP = 8
GLOBAL GL.COLOR.MATERIAL : GL.COLOR.MATERIAL = 2903
GLOBAL GL.LIGHTING : GL.LIGHTING = 2896
GLOBAL GL.LIGHT0 : GL.LIGHT0 = 16384
GLOBAL GL.LIGHT1 : GL.LIGHT1 = 16385
GLOBAL GL.LIGHT2 : GL.LIGHT2 = 16386
GLOBAL GL.LIGHT3 : GL.LIGHT3 = 16387
GLOBAL GL.LIGHT4 : GL.LIGHT4 = 16388
GLOBAL GL.LIGHT5 : GL.LIGHT5 = 16389
GLOBAL GL.LIGHT6 : GL.LIGHT6 = 16390
GLOBAL GL.LIGHT7 : GL.LIGHT7 = 16391
GLOBAL GL.NORMALIZE : GL.NORMALIZE = 2977
GLOBAL GL.TEXTURE.2D : GL.TEXTURE.2D = 3553
GLOBAL SDL.OPENGL : SDL.OPENGL = 2
GLOBAL SDL.INITVIDEO : SDL.INITVIDEO = 32
GLOBAL SDL.NOFRAME : SDL.NOFRAME = 32
' Main window size and location
WindowWidth = DisplayWidth : WindowHeight = DisplayHeight-30
UpperLeftX = 0 : UpperLeftY = 0
' 3D window size and location
GLwidth = DisplayWidth - 86 : GLheight = DisplayHeight - 104
UpperLeftGLx = 65 : UpperLeftGLy = 28
' Put a frame around the 3d display window
GRAPHICBOX #main.frame, UpperLeftGLx - 2 , UpperLeftGLy - 2 , GLwidth + 4 , GLheight + 4
BUTTON #main.close,"Close",[quit], UL , 5 , 26 , 55 , 30
OPEN "3D Viewer" FOR window AS #main
#main , "trapclose [quit]"
lHwnd = HWND( #main )
CALL LocateWindow lHwnd , UpperLeftX , UpperLeftY , WindowWidth , WindowHeight
' Set the initial rotation of the scene
xrot = 0 : yrot = 0 : zrot = 0
' Set camera location and viewing angle
eyeX = 0 : eyeY = 0 : eyeZ = 5
centerX = 0 : centerY = 0 : centerZ = 0
upX = 0 : upY = 5 : upZ = 0
CALL StartGL GLwidth , GLheight , "3d" , UpperLeftGLx , UpperLeftGLy
' Initialize a few 3D settings
CALL glClearColor 1 , 1 , 1 , 1
CALL glEnable GL.COLOR.MATERIAL
CALL glEnable GL.NORMALIZE
CALL glEnable GL.LIGHTING
CALL glEnable GL.LIGHT0
CALL ClearView eyeX , eyeY , eyeZ , centerX , centerY , centerZ , upX , upY , upZ
CALL RefreshView
RETURN
'XXXXXXXXXXXXXXXXXXXXXXXXX DO NOT ALTER ANYHING BELOW THIS LINE
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
[quit]
CALLDLL #sdl , "SDL_Quit" ,_
ret AS long
CLOSE #glu
CLOSE #gl
CLOSE #sdl
CLOSE #oleaut32
CLOSE #main
END
SUB glBegin mode' Begin the specified OpenGL mode
CALLDLL #gl , "glBegin" ,_
mode AS long ,_
ret AS void
END SUB
SUB glEnd' End the current mode
CALLDLL #gl , "glEnd" ,_
glEnd AS long
END SUB
SUB glEnable code' Enables the specified code
CALLDLL #gl , "glEnable" ,_
code AS long ,_
glEnable AS long
END SUB
SUB glDisable code' Disables the specified code
CALLDLL #gl , "glDisable" ,_
code AS long ,_
glDisable AS long
END SUB
SUB glColor4fv red , green , blue , alpha' sets the color for objects to be created
STRUCT glColor , red AS ulong , green AS ulong , blue AS ulong , alpha AS ulong
glColor.red.struct = R4( red )
glColor.green.struct = R4( green )
glColor.blue.struct = R4( blue )
glColor.alpha.struct = R4( alpha )
CALLDLL #gl , "glColor4fv" ,_
glColor AS STRUCT ,_
ret AS void
END SUB
```

```
SUB glMatrixMode mode
CALLDLL #gl , "glMatrixMode" ,_
mode AS long ,_
glMatrixMode AS long
END SUB
SUB glLoadIdentity' Loads the current identity into the matrix
CALLDLL #gl , "glLoadIdentity" ,_
glLoadIdentity AS long
END SUB
SUB gluPerspective fovy , aspect , zNear , zFar
CALLDLL #glu , "gluPerspective" ,_
fovy AS double ,_
aspect AS double ,_
zNear AS double ,_
zFar AS double ,_
gluPerspective AS void
END SUB
SUB glScalef xScale , yScale , zScale
xScale = R4( xScale )
yScale = R4( yScale )
zScale = R4( zScale )
CALLDLL #gl , "glScalef" ,_
xScale AS ulong ,_
yScale AS ulong ,_
zScale AS ulong ,_
glScalef AS long
END SUB
SUB glClearColor red , green , blue , alpha' clears the scene to the specified color
red = R4( red )
green = R4( green )
blue = R4( blue )
alpha = R4( alpha )
CALLDLL #gl , "glClearColor" ,_
red AS ulong ,_
green AS ulong ,_
blue AS ulong ,_
alpha AS ulong ,_
glClearColor AS long
END SUB
SUB ClearView eyeX , eyeY , eyeZ , centerX , centerY , centerZ , upX , upY , upZ
flag = 256 OR 16384
CALL glClear flag
CALL glMatrixMode 5888
CALL glLoadIdentity
CALL gluLookat eyeX , eyeY , eyeZ , centerX , centerY , centerZ , upX , upY , upZ
END SUB
```

```
SUB glClear code' Clears the buffer to the last specified color
CALLDLL #gl , "glClear" ,_
code AS long ,_
glClear AS long
END SUB
SUB gluLookat eyeX , eyeY , eyeZ , centerX , centerY , centerZ , upX , upY , upZ
CALLDLL #glu , "gluLookAt" ,_
eyeX AS double ,_
eyeY AS double ,_
eyeZ AS double ,_
centerX AS double ,_
centerY AS double ,_
centerZ AS double ,_
upX AS double ,_
upY AS double ,_
upZ AS double ,_
gluLookat AS void
END SUB
SUB glRotatef amt , x , y , z' rotates the scene around the specified axis
amt = R4( amt )
x = R4( x )
y = R4( y )
z = R4( z )
CALLDLL #gl , "glRotatef" ,_
amt AS ulong ,_
x AS ulong ,_
y AS ulong ,_
z AS ulong ,_
glRotatef AS void
END SUB
SUB RefreshView
CALLDLL #gl , "glFlush" ,_' Tells OpenGL to finish any calls that have been made
r AS void
CALLDLL #sdl , "SDL_GL_SwapBuffers" ,_' Swap the buffers to display what has been drawn since the
last swap
RefreshView AS void
END SUB
SUB glNormal x1 , y1 , z1 , x2 , y2 , z2 , x3 , y3 , z3' calcualtes the normal vector for an object,
Vx1 = x2 - x1 ' basically this is used so an object gets "lit up"
Vy1 = y2 - y1
Vz1 = z2 - z1
Vx2 = x3 - x1
Vy2 = y3 - y1
Vz2 = z3 - z1
nX = ( Vy2 * Vz1 ) - ( Vz2 * Vy1 )
nY = ( Vz2 * Vx1 ) - ( Vx2 * Vz1 )
```

```
nZ = ( Vx2 * Vy1 ) - ( Vy2 * Vx1 )
CALLDLL #gl , "glNormal3d" ,_
nX AS double ,_
nY AS double ,_
nZ AS double ,_
ret AS void
END SUB
SUB glVertex x , y , z
CALLDLL #gl , "glVertex3d" ,_
x AS double ,_
y AS double ,_
z AS double ,_
glVertex AS long
END SUB
SUB glLineWidth width
width = R4( width )
CALLDLL #gl , "glLineWidth" ,_
width AS ulong ,_
ret AS void
END SUB
SUB glPointSize width
width = R4( width )
CALLDLL #gl , "glPointSize" ,_
width AS ulong ,_
ret AS void
END SUB
SUB glGenLists num' reserves memory for specified number of call lists
CALLDLL #gl , "glGenLists" ,_
num AS long ,_
glGenLists AS long
END SUB
SUB glNewList list , mode' begin creating the specified CallList
CALLDLL #gl , "glNewList" ,_
list AS long ,_
mode AS long ,_
ret AS void
END SUB
SUB glEndList' end creating current CallList
CALLDLL #gl , "glEndList" ,_
glEndList AS long
END SUB
SUB glCallList listnum' tells OpenGL to send the specified CallList to the scene
CALLDLL #gl , "glCallList" ,_
listnum AS long ,_
glCallList AS long
END SUB
```

```
SUB glDeleteLists start , range' deletes the specified lists, starting from "start" through "range"
CALLDLL #gl , "glDeleteLists" ,_
start AS long ,_
range AS long ,_
ret AS void
END SUB
SUB StartGL WinWidth , WinHeight , WinCaption$ , ULX , ULY
OPEN "SDL.dll" FOR dll AS #sdl
OPEN "opengl32.dll" FOR dll AS #gl
OPEN "glu32.dll" FOR dll AS #glu
OPEN "oleaut32" FOR DLL AS #oleaut32
CALLDLL #sdl , "SDL_Init" ,_
SDL.INITVIDEO AS long ,_
result AS long
flag = SDL.NOFRAME OR SDL.OPENGL
CALLDLL #sdl , "SDL_SetVideoMode" ,_
WinWidth AS long ,_
WinHeight AS long ,_
SDL.INITVIDEO AS long ,_
flag AS long ,_
ecran AS long
CALLDLL #sdl , "SDL_WM_SetCaption" ,_
WinCaption$ AS ptr ,_
"" AS ptr ,_
r AS long
lpWindowName$ = "3d"
CALLDLL #user32 , "FindWindowA" ,_
0 AS long ,_
lpWindowName$ AS ptr ,_
fHwnd AS ulong
hwnd = HWND( #main )
CALLDLL #user32 , "SetParent" ,_
fHwnd AS ulong ,_
hwnd AS ulong ,_
result AS long
UpperLeftX = ULX
UpperLeftY = ULY
WindowWidth = WinWidth
WindowHeight = WinHeight
CALL LocateWindow fHwnd , UpperLeftX , UpperLeftY , WindowWidth , WindowHeight
CALL glEnable 2929
CALL glMatrixMode 5889
CALL glLoadIdentity
CALL gluPerspective 45 , WinWidth / WinHeight , .01 , 100
END SUB
SUB LocateWindow hWnd , ULx , ULy , Width , Height
```

```
CALLDLL #user32 , "MoveWindow" ,_
hWnd AS ULong ,_
ULx AS Long ,_
ULy AS Long ,_
Width AS Long ,_
Height AS Long ,_
1 AS boolean ,_
success AS Long
END SUB
SUB CreateBMPTexture filename$ , texture' loads a bitmap in a form that can be used by OpenGL for
textures
call glBindTexture GL.TEXTURE.2D , texture
OPEN "DevIL.dll" FOR dll AS #dev
CALLDLL #dev , "ilInit" ,_
ret AS long
CALLDLL #dev , "ilGenImages" ,_
1 AS long ,_
ret AS long
CALLDLL #dev , "ilBindImage" ,_
1 AS long ,_
ret AS long
CALLDLL #dev , "ilLoadImage" ,_
filename$ AS ptr ,_
ret AS long
CALLDLL #dev , "ilConvertImage" ,_
6408 AS long ,_
5121 AS long ,_
ret AS long
CALLDLL #dev , "ilGetData" ,_
array AS long
STRUCT arr , bitmaparray AS long
arr.bitmaparray.struct = array
CALLDLL #dev , "ilGetInteger" ,_
3560 AS long ,_
IL.IMAGE.BPP AS long
CALLDLL #dev , "ilGetInteger" ,_
3556 AS long ,_
IL.IMAGE.WIDTH AS long
CALLDLL #dev , "ilGetInteger" ,_
3557 AS long ,_
IL.IMAGE.HEIGHT AS long
CALLDLL #dev , "ilGetInteger" ,_
3562 AS long ,_
IL.IMAGE.FORMAT AS long
CALLDLL #gl , "glTexImage2D" ,_
GL.TEXTURE.2D AS long ,_
```

```
0 AS long ,_
IL.IMAGE.BPP AS long ,_
IL.IMAGE.WIDTH AS long ,_
IL.IMAGE.HEIGHT AS long ,_
IL.IMAGE.FORMAT AS long ,_
5121 AS long ,_
array AS long ,_
ret AS long
CALLDLL #glu , "gluBuild2DMipmaps" ,_
GL.TEXTURE.2D AS long ,_
4 AS long ,_
IL.IMAGE.WIDTH AS long ,_
IL.IMAGE.HEIGHT AS long ,_
6408 AS long ,_
5121 AS long ,_
arr AS long ,_
ret AS long
CALLDLL #dev , "ilDeleteImages" ,_
1 AS long ,_
ret AS long
CLOSE #dev
END SUB
SUB glBindTexture target , texture
CALLDLL #gl , "glBindTexture" ,_
target AS long ,_
texture AS long ,_
ret AS void
END SUB
SUB TextureVertex s , t , x , y , z' defines the cooridinates for one vertex of a textured object
s = R4( s )
t = R4( t )
x = R4( x )
y = R4( y )
z = R4( z )
CALLDLL #gl , "glTexCoord2f" ,_
s AS long ,_
t AS long ,_
ret AS void
CALLDLL #gl , "glVertex3f" ,_
x AS long ,_
y AS long ,_
z AS long ,_
ret AS void
END SUB
SUB glGenTextures num
STRUCT glGenTextures ,_
```

```
glGenTexture1 AS long ,_
glGenTexture2 AS long ,_
glGenTexture3 AS long ,_
glGenTexture4 AS long ,_
glGenTexture5 AS long ,_
glGenTexture6 AS long ,_
glGenTexture7 AS long ,_
glGenTexture8 AS long ,_
glGenTexture9 AS long ,_
glGenTexture10 AS long ,_
glGenTexture11 AS long ,_
glGenTexture12 AS long ,_
glGenTexture13 AS long ,_
glGenTexture14 AS long ,_
glGenTexture15 AS long ,_
glGenTexture16 AS long ,_
glGenTexture17 AS long ,_
glGenTexture18 AS long ,_
glGenTexture19 AS long ,_
glGenTexture20 AS long
CALLDLL #gl , "glGenTextures" ,_
num AS long ,_
glGenTextures AS STRUCT ,_
ret AS void
END SUB
SUB Pause tyme
CALLDLL #kernel32, "Sleep" ,_
tyme AS long ,_
re AS void
END SUB
FUNCTION R4( R8 )' used to convert a number to a float for some of the OpenGL calls, Credit to Brent Thorn
STRUCT local1 ,_
R4 AS ULong
CALLDLL #oleaut32 , "VarR4FromR8" ,_
R8 AS Double ,_
local1 AS STRUCT ,_
ret AS Long
R4 = local1.R4.struct
END FUNCTION
```