日本図学会東北支部講演会 10/17/1998

Fortranによる3次元コンピュータ・グラフィックスOpenGLについて

いわき明星大学　五十嵐　三武郎、 露木　浩二

1　緒言

流れの数値シミュレーションがハードウエアの進歩と性能向上によって広い分野で応用されている。それに伴って得られた結果を用いて現象の解明が重要となっている。その1つにコンピュータビジュアライゼイションがある。これまで2次元の現象が多くシミュレーションされてきたが計算機の性能向上に伴って3次元現象のシミュレーションが多く行われるようになってきている。その結果、3次元グラフィックスが重要な役割を果たすようになっている。具体的には数値シミュレーションは高速のコンパイラのあるFortran で行い visualization は高価な市販のソフトを用いる。このときソフトに適合したデータ変換の作業や変換されたデータの確保のため多くの資源が必要とされる。通常数値計算結果は Fortran の書式なしで出力したほうがファイルサイズが4分の1程度にはなる。したがって計算結果のファイルを直接読み込んで視覚化することが重要となってくる。本研究ではこの点に着目して3次元のグラフィックスが可能でFortran で記述できるものとして OpenGL を用いてグラフィックスを行う。

OpenGL なるグラフィックスは UNIX をはじめとして多くのプラットフォーム, WINDOWS 95/NT あるいは MAC にものっている。しかしもともと C 言語の使用を主としているので Fortran での使用については十分な情報が見あたらない。また最終的な画面表示は低水準のソフトウエアであり OpenGL はそれより高水準に位置している。この部分が OS によって異なっている。本稿では Windows 95/NT 上で動作する Microsoft(DEC) の Visual Fortran のもとでの OpenGL を用いて運動する3次元物体のコンピュータ・グラフィックスを示す。

2　Visual Fortranについて

Fortran は Fortan90 の規格に基づいたコンパイラーで各社によって種々の拡張がある。ここではもともと DEC 社の開発した DEC Fortran に Microsoft の Fortran を合体した Visual Fortran を用いる。この場合基本的なグラフィックスは qwins というライブラリーの形で提供されている。ユーザーは OpenGL を用いるときはこの qwins の内容について知る必要はない。OpenGL から呼ばれる形で使われている。Fortran90 は Fortran77 の拡張版であるが、主な特徴は次の通りである。

1. 自由書式の導入、これによってこれまでのカードイメージのソースファイルの作成は不要となった。
2. use 文を用いてあらかじめコンパイルしたモジュールを使える。
3. コメントは ！ なる文字で始まるそれ以降の文はコメントとなる。したがって従来のように1行目に ！ を書いてもよいし、

   x = y + z ！これ以降コメント

   のように文の後にもおける。
4. サブルーチンの RETURN、END の代わりに RETURN の不要な

   END　SUBROUTINE　サブルーチン名

   の1行でもよい。
5. 論理判定の .EQ. などのほかに ==、> 0、>= 0 等が許される。
6. 行列演算のためのベクトル関数の導入がはかられている。
7. 変数の型を自由に作れる。type というキーワードを用いて C 言語の structure、Pascal の record のようにできる。
8. continuation は文中に & の記号を用いて F77 の6カラムでの制限をなくしている。
9. 全体的には限りなく C 言語や Delphi に近づいている感がある。

したがってC言語の見本があればFortranへの移行は難しくない。OpenGLをFortranで引用するにはC言語でglで始まる関数を
fglに変えればよい。Cの場合の

```
gl...;
```

は

```
call fgl....
```

となる。

3 静止した長方形

3.1 ソースプログラム

長方形を書いて内部を色塗りするプログラムである。

```
! SimpleRect.f90
use dfopngl   ! DEC Fortran の OpenGL
use dfwin     ! DEC Visual Fortran の Graphics

integer(4)      ret
integer(2)      pattern

    call fauxInitDisplayMode (IOR(AUX_SINGLE , AUX_RGB))
    call fauxInitPosition (50, 50, 400, 400)
    ret = fauxInitWindow ("Simple Rectangle by Igarashi")
    call fglClearColor (0.0, 0.0, 0.0, 0.0)
    call fglClear(GL_COLOR_BUFFER_BIT)
    call fglColor3f(1.0, 1.0, 1.0)
    call fglLoadIdentity ()
    !
    ! Drawing Rect
    !
    call fglColor3f (1.0, 0.0, 0.0)
    call fglBegin(GL_POLYGON)
        call fglVertex2f(-0.25, -0.25)
        call fglVertex2f(0.25, -0.25)
        call fglVertex2f(0.25, 0.25)
        call fglVertex2f(-0.25, 0.25)
    call fglEnd()
    call fglFlush()
    call sleep (2000)
end
```

3.2 コンパイルとリンク

DECの説明書はFortanプログラムに対してnmakeというC言語で良く用いられる手法を用いてコンパイルとリンクを行っている。しかしほとんどのFortranのユーザーはコマンドラインから

```
f77....
```

のように行うのが普通である。そこで基本的にどうなっているか調べたところ次のようにすればようことがわかった。上記の Fortran90 のプログラム名を simple.f90 とする。

3.2.1 コンパイル

> df /c simple.f90 /libs:qwin

DEC Fortran のコンパイラー DF を呼び出す。その際の option は/c はコンパイルのみで simple.obj のみを作る。/libs : qwin はグラフィックスライブラリーの指定である。

3.2.2 リンク

> link advapi32.lib simple.obj

としてリンクすべきライブラリ advapi32.lib を指定する。すると console について warning がでるが問題ない。結果として simple.exe が最終的にできあがり実行すればよい。

3.3 結果

simple.exe を実行すると 1 つの Window の画面が開いてそこにグラフィックスが表示される。なおこの表示は指定の時間が経つと消えるようになっている。Windows 特有のプログラムを作るには画面サイズ、位置の変更など行っても動くようにする必要がある。

一般的な運動する場合、回転する 2 次元長方形と回転する 3 つの歯車が種々の座標軸まわりのまわりに回転する場合のプログラムを付録に示す。

4 結論

Fortran による OpenGL を用いて 3 次元グラフィックスを行う基礎的な研究を行った。まず 2 次元の静止した長方形と回転する長方形を扱った。ついで回転する 3 次元歯車の運動をキーボードやマウスの操作にしたっがて反応するプログラムを作成した。

なおここに述べたプログラムはC言語で OpenGL の文献や OpenGL にほぼコンパチの Mesa というソフトウエアを参考にして Fortran で記述したものである。このプログラムは X86 の CPU に対するもので、DEC の alpha に対しては若干の修正が必要である。

本研究は「流れのビジュアライゼイション －数値シミュレーションと実験－」という研究課題で平成 9 年度いわき明星大学特別研究助成費 (A) を受けて行いました。ここに謝意を表します。

参考文献

[1] 五十嵐、露木、対馬 : OpenGL によるコンピュータ・グラフィックス I, 日本図学会東北支部講演会 1998 年 3 月, 東北工業大学にて.

[2] 三浦 憲二郎 : OpenGL 3D グラフィックス入門、朝倉書店、 1995.

[3] J. Neider, T.Davis and M. Woo : OpenGL Programing Guide, the Official Guide to Learning OpenGL, Release 1, Addison- Wesley Pub. Co., 1993.

[4] 床井 浩平: AUX ライブラリを使った手抜き OpenGL 入門、http://www.center.wakayama-u.ac.jp/~tokoi/index-html.

[5] Brian Paul:Mesa の作者、http://www.ssec.wisc.edu/~brianp/home.html

付録１：回転する長方形

ソースプログラム

```
!/* spin0.f90 */
! 新しい fortran90 規格による：contains 文による subroutine の記述
!  main に書いた変数は大域的になる。
!  従来の場合には次の点に注意
!  ********************************************************
!  pointer による subroutine の引用の LOC の場合には
!  subroutine の次に  !DEC$  の directive をおく
!  その subroutine が  main  の外部で定義されているときには
!    main  program  の  use 文の後に
!    interface  ...    end  interface  で定義し
!    main  program の使用する変数宣言へと続くようにする。
!  ********************************************************
!
! Draw a rectangle
program main
use dfwin
use dfopngl

integer(4) ret
integer(4) type
real(4)  :: spin=0.0

   call fauxInitPosition (0, 0, 300, 300)
   type= AUX_RGB
   type= ior(type, AUX_DOUBLE)
   call fauxInitDisplayMode( TYPE)

   ret = fauxInitWindow("Rectangle by Fortran")

   call init()


   call fauxExposeFunc( LOC(Reshape)) !あってもなくてもよい
   call fauxReshapeFunc( LOC(Reshape) )

     call fauxKeyFunc(AUX_LEFT, LOC(left))    !;//左矢印
!    call fauxKeyFunc(AUX_RIGHT, LOC(right))  ! ;//右矢印

!    call fauxKeyFunc(AUX_UP, LOC(up))        !;//上矢印
!    call fauxKeyFunc(AUX_DOWN, LOC(down))    !;//下矢印
!    call fauxKeyFunc(AUX_X, LOC(Space_key) )     !;//大文字 X キー
!    call fauxKeyFunc(AUX_Z,LOC(Z_key))       !;//大文字Ｚキー
```

```
    call fauxIdleFunc( LOC(SpinDisplay) )
    call fauxMainLoop(NULL)
!
    Contains
!
! all set of subroutines contained in Main

subroutine  draw( )
use dfopngl ! comment でもよい

   call fglClear( ior(GL_COLOR_BUFFER_BIT, GL_DEPTH_BUFFER_BIT) )
   call fglPushMatrix()
   call fglRotatef( spin, 0.0, 0.0, 1.0 )
   call fglRectf(-25.0,-25.0,25.0,25.0)
   call fglPopMatrix()
   call fglFlush()

   call fauxSwapBuffers()      !フレームバッファの交換
end subroutine draw

subroutine spinDisplay(  )
!DEC$ ATTRIBUTES STDCALL, ALIAS : '_Spindiaplay@0' :: Spindisplay

  spin = spin + 2.0
  if(spin > 360.0) spin = spin - 360.0
  if(spin < 0.0 )  spin = spin + 360.0
  call draw()
end subroutine spinDisplay

subroutine Right()
!DEC$ ATTRIBUTES STDCALL, ALIAS : '_Right@0' :: Right

    spin = spin + 2.0
end subroutine right

subroutine Left()
!DEC$ ATTRIBUTES STDCALL, ALIAS : '_Left@0' :: Left

    spin = spin - 2.0
end subroutine left

!/* new window size or exposure */
subroutine reshape(  width,  height )
!DEC$ ATTRIBUTES STDCALL, ALIAS : '_Reshape@8' :: Reshape
```

```
use dfopngl  !comment でもよい

integer(4) width, height

   call fglViewport(0, 0, width, height)
   call fglMatrixMode(GL_PROJECTION)
   call fglLoadIdentity()
   if (width>height) then
      w = real( width )/ real( height)
      call fglOrtho( -50.0*w,50.0*w, -50.0, 50.0, -1.0, 1.0 )
   else
      h = real( height) / real( width )
      call fglOrtho( -50.0, 50.0, -50.0*h, 50.0*h, -1.0, 1.0 )
   end if

   call fglMatrixMode(GL_MODELVIEW)
   call fglLoadIdentity()
end subroutine Reshape

subroutine init( )
use dfopngl !comment でもよい

   call fglClearColor(0.0,0.0,0.0,0.0)
   call fglColor3f(1.0,1.0,1.0)
   call fglShadeModel(GL_FLAT)
end subroutine init

end program main
```

付録 2:回転する歯車

ソースプログラム

```
!/* gears0.f90 */

!/*
! * 3-D gear wheels.  This program is in the public domain.
! *
! * Brian Paul
! *
!  Translated from C++
!  NO inc file

Program main
use dfopngl
```

```
use dfwin

! interfgace
! interface 部不要: Contains
! end interface

!main start

!Global variable:
real, parameter :: M_PI = 3.141592654
real(4) view_rotx,view_roty,view_rotz,rot_angle
integer ret,type
integer gear1,gear2,gear3

! Initial value defined in gearsINC module
   view_rotx=20.0
   view_roty=30.0
   view_rotz=0.0
   rot_angle = 0.0

   type=AUX_RGB
   type=IOR(type,AUX_DEPTH)
   type=IOR(type,AUX_DOUBLE)
   type=IOR(type,AUX_DIRECT)

   call fauxInitPosition (0, 0, 300, 300)
   call fauxInitDisplayMode(type)
!   ret= fauxInitWindow("回転 Gears、矢印キー、Z")
   ret= fauxInitWindow("Fortran Gears by S. I.")

   call init()
   call fauxExposeFunc( LOC(Reshape)) !あってもなくてもよい
   call fauxReshapeFunc( LOC(reshape) )

!    /* マウスによる移動 */
!   // auxMouseFunc(AUX_LEFTBUTTON, AUX_MOUSEDOWN, addArm1);
!   // auxMouseFunc(AUX_RIGHTBUTTON, AUX_MOUSEDOWN, subtractArm1);
!    /* キーによる移動 */
    call fauxKeyFunc(AUX_LEFT, LOC(left))    !;//左矢印
    call fauxKeyFunc(AUX_RIGHT, LOC(right))  ! ;//右矢印
    call fauxKeyFunc(AUX_UP, LOC(up))        !;//上矢印
    call fauxKeyFunc(AUX_DOWN, LOC(down))    !;//下矢印
    call fauxKeyFunc(AUX_X, LOC(Space_key) )     !;//大文字 X キー
    call fauxKeyFunc(AUX_Z,LOC(Z_key))       !;//大文字Ｚキー
    call fauxIdleFunc( LOC(idle) )           !;//回転運動
    call fauxMainLoop( NULL )                !;//dummy だが運動表示のため必要
```

```
   CONTAINS

!/*
! * Draw a gear wheel.  You'll probably want to call this function when
! * building a display list since we do a lot of trig here.
! *
! * Input:  inner_radius - radius of hole at center
! *         outer_radius - radius at center of teeth
! *         gwidth - width of gear
! *         teeth - number of teeth
! *         tooth_depth - depth of tooth
! */


subroutine gear(inner_radius, outer_radius, gwidth,teeth, tooth_depth )
use dfopngl
use dfwin

real  inner_radius,outer_radius, tooth_depth
integer  teeth
integer i
real  r0, r1, r2
real angle, da
real u, v, len

   r0 = inner_radius
   r1 = outer_radius - tooth_depth/2.0
   r2 = outer_radius + tooth_depth/2.0

   da = 2.0*M_PI / teeth / 4.0

   call fglShadeModel( GL_FLAT )

   call fglNormal3f( 0.0, 0.0, 1.0 )

!   /* draw front face */
   call fglBegin( GL_QUAD_STRIP )
   do i=0,teeth
      angle = i * 2.0*M_PI / teeth
      call fglVertex3f( r0*cos(angle), r0*sin(angle),gwidth*0.5 )
      call fglVertex3f( r1*cos(angle), r1*sin(angle),gwidth*0.5 )
      call fglVertex3f( r0*cos(angle), r0*sin(angle),gwidth*0.5 );
      call fglVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da), gwidth*0.5 )
    end do
   call fglEnd()
```

```
!   /* draw front sides of teeth */
    call fglBegin( GL_QUADS )
    da = 2.0*M_PI / teeth / 4.0
    do i=0,teeth
       angle = i * 2.0*M_PI / teeth

       call fglVertex3f(r1*cos(angle),r1*sin(angle),gwidth*0.5 )
       call fglVertex3f(r2*cos(angle+da),r2*sin(angle+da),gwidth*0.5 )
       call fglVertex3f(r2*cos(angle+2*da),r2*sin(angle+2*da),gwidth*0.5 )
       call fglVertex3f(r1*cos(angle+3*da),r1*sin(angle+3*da),gwidth*0.5 )
     end do
    call fglEnd()


    call fglNormal3f( 0.0, 0.0, -1.0 )

!   /* draw back face */
    call fglBegin( GL_QUAD_STRIP )
    do i=0,teeth
       angle = i * 2.0*M_PI / teeth
       call fglVertex3f( r1*cos(angle), r1*sin(angle), -gwidth*0.5 )
       call fglVertex3f( r0*cos(angle), r0*sin(angle), -gwidth*0.5 )
       call fglVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da), -gwidth*0.5 )
       call fglVertex3f( r0*cos(angle), r0*sin(angle), -gwidth*0.5 )
     end do
    call fglEnd()

!   /* draw back sides of teeth */
    CALL FglBegin( GL_QUADS )
    da = 2.0*M_PI / teeth / 4.0
    do i=0,teeth
       angle = i * 2.0*M_PI / teeth

       call fglVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da), -gwidth*0.5 )
       call fglVertex3f( r2*cos(angle+2*da), r2*sin(angle+2*da), -gwidth*0.5 )
       call fglVertex3f( r2*cos(angle+da),   r2*sin(angle+da),   -gwidth*0.5 )
       call fglVertex3f( r1*cos(angle),      r1*sin(angle),      -gwidth*0.5 )
     end do
    call fglEnd()

!   /* draw outward faces of teeth */
    call fglBegin( GL_QUAD_STRIP )
    do i=0,teeth
       angle = i * 2.0*M_PI / teeth
```

```
      call fglVertex3f( r1*cos(angle), r1*sin(angle), gwidth*0.5 )
      call fglVertex3f( r1*cos(angle), r1*sin(angle), -gwidth*0.5 )
      u = r2*cos(angle+da) - r1*cos(angle)
      v = r2*sin(angle+da) - r1*sin(angle)
      len = sqrt( u*u + v*v )
      u = u/ len
      v = v/ len
      call fglNormal3f( v, -u, 0.0 )
      call fglVertex3f( r2*cos(angle+da), r2*sin(angle+da), gwidth*0.5 )
      call fglVertex3f( r2*cos(angle+da), r2*sin(angle+da), -gwidth*0.5 )
      call fglNormal3f( cos(angle), sin(angle), 0.0 )
      call fglVertex3f( r2*cos(angle+2*da), r2*sin(angle+2*da),gwidth*0.5 )
      call fglVertex3f( r2*cos(angle+2*da), r2*sin(angle+2*da), -gwidth*0.5)
      u = r1*cos(angle+3*da) - r2*cos(angle+2*da)
      v = r1*sin(angle+3*da) - r2*sin(angle+2*da)
      call fglNormal3f( v, -u, 0.0 )
      call fglVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da),gwidth*0.5)
      call fglVertex3f( r1*cos(angle+3*da), r1*sin(angle+3*da), -gwidth*0.5)
      call fglNormal3f( cos(angle), sin(angle), 0.0 )
    end do

   call fglVertex3f( r1*cos(0.), r1*sin(0.), gwidth*0.5 )
   call fglVertex3f( r1*cos(0.), r1*sin(0.), -gwidth*0.5 )

   call fglEnd()


   call fglShadeModel( GL_SMOOTH )

!   /* draw inside radius cylinder */
   CALL FglBegin( GL_QUAD_STRIP )
   do i=0,teeth
      angle = i * 2.0*M_PI / teeth
      call fglNormal3f( -cos(angle), -sin(angle), 0.0 )
      call fglVertex3f( r0*cos(angle), r0*sin(angle), -gwidth*0.5 )
      call fglVertex3f( r0*cos(angle), r0*sin(angle), gwidth*0.5 )
   end do
   call fglEnd()

end subroutine gear

subroutine draw()
use dfopngl
use dfwin

! //glClearColor(0.0,0.0,0.0,0.0);
```

```
!       //background color= BLACK(default):大画面では見にくいので白にするには次行のように
する

   call fglClearColor(1.0,1.0,1.0,1.0)
!   //backgroundcolor=white: DOS 大画面でよく見えるように

   call fglClearColor(0.0,1.0,1.0,0.0)
!   //backgroundcolor=green: windows 画面のように

   call fglClear( IOR(GL_COLOR_BUFFER_BIT, GL_DEPTH_BUFFER_BIT) )

   call fglPushMatrix()
   call fglRotatef( view_rotx, 1.0, 0.0, 0.0 )
   call fglRotatef( view_roty, 0.0, 1.0, 0.0 )
   call fglRotatef( view_rotz, 0.0, 0.0, 1.0 )

   call fglPushMatrix()
   call fglTranslatef( -3.0, -2.0, 0.0 )
   call fglRotatef( rot_angle, 0.0, 0.0, 1.0 )
   call fglCallList(gear1)
   call fglPopMatrix()

   call fglPushMatrix()
   call fglTranslatef( 3.1, -2.0, 0.0 )
   call fglRotatef( -2.0*rot_angle-9.0, 0.0, 0.0, 1.0 )
   call fglCallList(gear2)
   call fglPopMatrix()

   call fglPushMatrix()
   call fglTranslatef( -3.1, 4.2, 0.0 )
   call fglRotatef( -2.0*rot_angle-25.0, 0.0, 0.0, 1.0 )
   call fglCallList(gear3)
   call fglPopMatrix()

   call fglPopMatrix()

  call fauxSwapBuffers()

end subroutine draw

subroutine idle( )
!DEC$ ATTRIBUTES STDCALL, ALIAS : '_idle@0' :: idle

use dfwin

   rot_angle = rot_angle + 2.0
```

```
   call draw()
end subroutine idle


!void CALLBACK none(void)
!{
!}


subroutine Right()
!DEC$ ATTRIBUTES STDCALL, ALIAS : '_Right@0' :: Right

  view_roty =  view_roty - 5.0

end subroutine right

subroutine Left()
!DEC$ ATTRIBUTES STDCALL, ALIAS : '_Left@0' :: Left

  view_roty = view_roty + 5.0

end subroutine left

subroutine Up()
!DEC$ ATTRIBUTES STDCALL, ALIAS : '_Up@0' :: Up

  view_rotx = view_rotx + 5.0

end subroutine up

subroutine Down()
!DEC$ ATTRIBUTES STDCALL, ALIAS : '_Down@0' :: Down

  view_rotx = view_rotx - 5.0

end subroutine down

subroutine Space_key()
!DEC$ ATTRIBUTES STDCALL, ALIAS : '_SPACE_Key@0' :: SPACE_Key

   view_rotz = view_rotz - 5.0
end subroutine Space_key

subroutine Z_key()
!DEC$ ATTRIBUTES STDCALL, ALIAS : '_Z_Key@0' :: Z_Key
```

```
  view_rotz  = view_rotz + 5.0

end subroutine Z_key


!/* new window size or exposure */
subroutine reshape( width, height )
!DEC$ ATTRIBUTES STDCALL, ALIAS : '_Reshape@8' :: Reshape

use dfopngl
use dfwin

integer width,height
real w,h

   call fglViewport(0, 0, width, height)
   call fglMatrixMode(GL_PROJECTION)
   call fglLoadIdentity()
   if (width .gt. height) then
      w = real( width) / real( height)
      call fglFrustum( -w, w, -1.0, 1.0, 5.0, 60.0 )
   else
      h = real( height) / real(width)
      call fglFrustum( -1.0, 1.0, -h, h, 5.0, 60.0 )
   end if

   call fglMatrixMode(GL_MODELVIEW)
   call fglLoadIdentity()
   call fglTranslatef( 0.0, 0.0, -40.0 )
   call fglClear( IOR(GL_COLOR_BUFFER_BIT, GL_DEPTH_BUFFER_BIT) )

end subroutine reshape

subroutine init( )
use dfopngl
use dfwin

   real pos(4),red(4),green(4),blue(4)

   data pos/5.0, 5.0, 10.0, 0.0/
   data red/0.8, 0.1, 0.0, 1.0/
   data green/0.0, 0.8, 0.2, 1.0/
   data blue/0.2, 0.2, 1.0, 1.0/

   call fglLightfv( GL_LIGHT0, GL_POSITION, LOC(pos) )
   call fglEnable( GL_CULL_FACE )
```

```
   call fglEnable( GL_LIGHTING )
   call fglEnable( GL_LIGHT0 )
   call fglEnable( GL_DEPTH_TEST )

!   /* make the gears */
   gear1 = fglGenLists(1)
   call fglNewList(gear1, GL_COMPILE)
   call fglMaterialfv( GL_FRONT, GL_AMBIENT_AND_DIFFUSE, LOC(red) )
   call gear( 1.0, 4.0, 1.0, 20, 0.7 )
   call fglEndList()

   gear2 = fglGenLists(1)
   call fglNewList(gear2, GL_COMPILE)
   call fglMaterialfv( GL_FRONT, GL_AMBIENT_AND_DIFFUSE, LOC(green) )
   call gear( 0.5, 2.0, 2.0, 10, 0.7 )
   call fglEndList()

   gear3 = fglGenLists(1)
   call fglNewList(gear3, GL_COMPILE)
   call fglMaterialfv( GL_FRONT, GL_AMBIENT_AND_DIFFUSE, LOC(blue) )
   call gear( 1.3, 2.0, 0.5, 10, 0.7 )
   call fglEndList()

   call fglEnable( GL_NORMALIZE )

end  subroutine init

end Program main
```