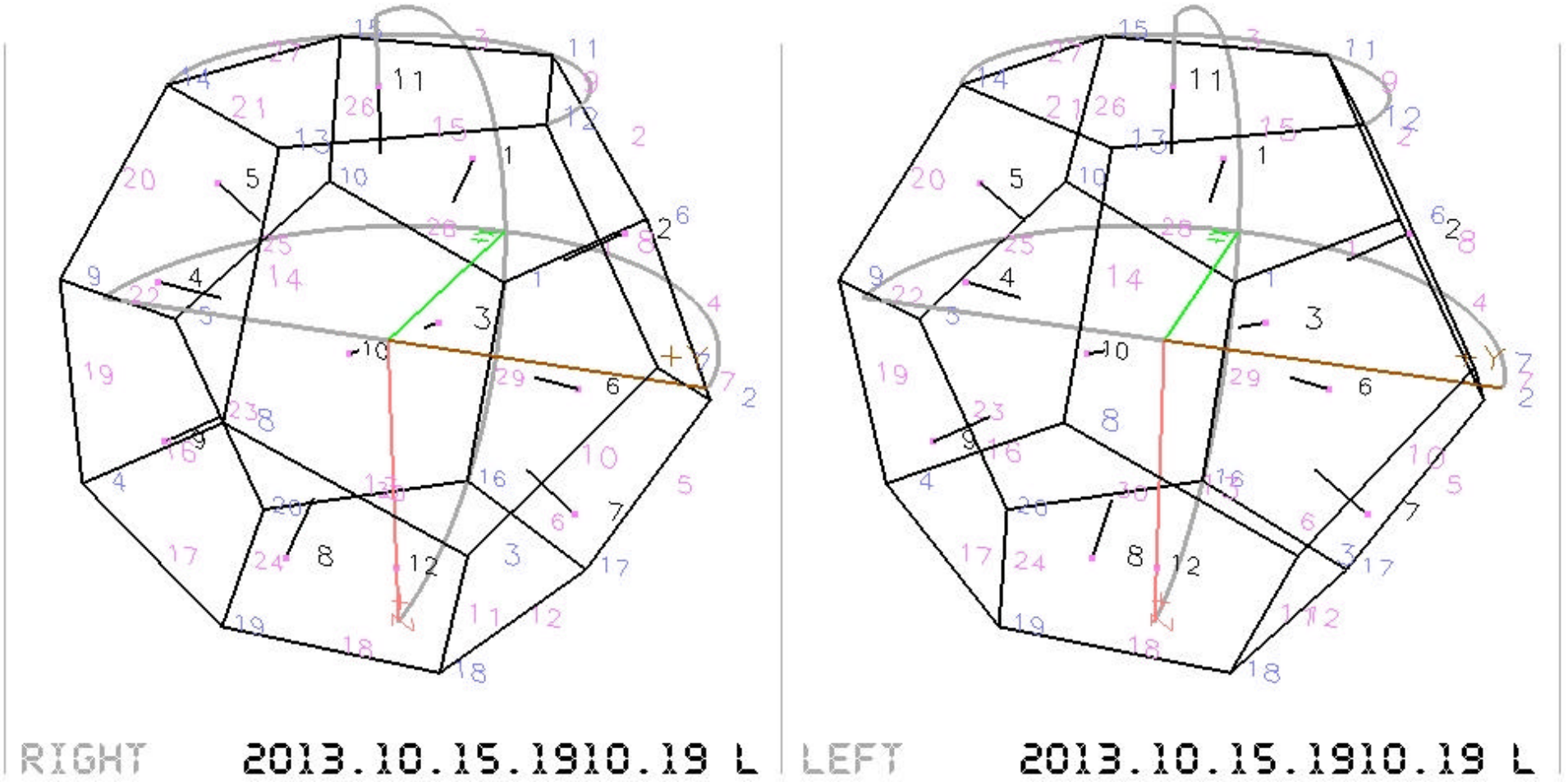
A Dodecahedron - the 12-face Platonic Solid – 3D views, Datae, & the Source Code.

On pages -> 1 & 8 2,3,& 7 4 & 5

- The vertices fit inside a sphere of radius=1. with geometry control based on using:
  Vertex #1 pitch angle = -10.81231696357170629128494477447210_16 degrees
  Vertex #16 pitch angle = -52.62263185935030435714286150500714_16 degrees
- The resulting vertices are regularly-spaced to the accuracy shown on page two.
- My reference: Wikipedia's in-depth article: "Dodecahedron". Theoretical values: see pages 6 & 7.

A crossed-eye 3D visualization of the “Setterholm Model” orientation & indices:



See also: http://ftp.setterholm.com/Fortran/Dodec-3D-RightLeft.jpg

The “Setterholm Model” indices:

Face:	CCW looking out Vertex Index
# 1=	1, 6,11,15,10/)
# 2=	2, 7,12,11, 6/)
# 3=	3, 8,13,12, 7/)
# 4=	4, 9,14,13, 8/)
# 5=	5,10,15,14, 9/)
# 6=	6, 1,16,17, 2/)
# 7=	7, 2,17,18, 3/)
# 8=	8, 3,18,19, 4/)
# 9=	9, 4,19,20, 5/)
#10=	10, 5,20,16, 1/)
#11=	13,14,15,11,12/)
#12=	19,18,17,16,20/)

Vertex:	CCW looking out Face Index
# 1=	1,10, 6/)
# 2=	2, 6, 7/)
# 3=	3, 7, 8/)
# 4=	4, 8, 9/)
# 5=	5, 9,10/)
# 6=	6, 2, 1/)
# 7=	7, 3, 2/)
# 8=	8, 4, 3/)
# 9=	9, 5, 4/)
#10=	10, 1, 5/)
#11=	1, 2,11/)
#12=	2, 3,11/)
#13=	3, 4,11/)
#14=	4, 5,11/)
#15=	5, 1,11/)
#16=	6,10,12/)
#17=	7, 6,12/)
#18=	8, 7,12/)
#19=	9, 8,12/)
#20=	10, 9,12/)

Edge:	Vertex Index &	ComputedLength
# 1=	1, 6/)	: 0.713644179546179863883939686092 67
# 2=	6,11/)	: 0.713644179546179863883939686092 16
# 3=	11,15/)	: 0.713644179546179863883939686092 88
# 4=	6, 2/)	: 0.713644179546179863883939686092 64
# 5=	2,17/)	: 0.713644179546179863883939686092 16
# 6=	17,16/)	: 0.713644179546179863883939686092 83
# 7=	2, 7/)	: 0.713644179546179863883939686090 14
# 8=	7,12/)	: 0.713644179546179863883939686092 16
# 9=	12,11/)	: 0.713644179546179863883939686091 28
#10=	7, 3/)	: 0.713644179546179863883939686092 64
#11=	3,18/)	: 0.713644179546179863883939686092 15
#12=	18,17/)	: 0.713644179546179863883939686091 26
#13=	3, 8/)	: 0.713644179546179863883939686092 66
#14=	8,13/)	: 0.713644179546179863883939686092 15
#15=	13,12/)	: 0.713644179546179863883939686092 83
#16=	8, 4/)	: 0.713644179546179863883939686092 64
#17=	4,19/)	: 0.713644179546179863883939686092 15
#18=	19,18/)	: 0.713644179546179863883939686092 83
#19=	4, 9/)	: 0.713644179546179863883939686092 67
#20=	9,14/)	: 0.713644179546179863883939686092 16
#21=	14,13/)	: 0.713644179546179863883939686092 82
#22=	9, 5/)	: 0.713644179546179863883939686090 12
#23=	5,20/)	: 0.713644179546179863883939686092 15
#24=	20,19/)	: 0.713644179546179863883939686091 27
#25=	5,10/)	: 0.713644179546179863883939686092 66
#26=	10,15/)	: 0.713644179546179863883939686092 15
#27=	15,14/)	: 0.713644179546179863883939686091 26
#28=	10, 1/)	: 0.713644179546179863883939686092 75
#29=	1,16/)	: 0.713644179546179863883939686092 15
#30=	16,20/)	: 0.713644179546179863883939686092 90

Theoretical Edge Length= 0.713644179546179863883939686092 17
=(sqrt(5.)-1.)/sqrt(3.)

1 2 3
12345678901234567890123456789012

RMS Error = 0.737926D-30 : ^

Here:
real*16 precision = 31 significant digits
& printouts are often zeros beyond ~ 34.

The “Setterholm Model” coordinates - version 1.0:

Vertices:	X	Y	Z
1*=(/	0.98224694637684602281567027523 5,	0.000000000000000000000000000000,	0.18759247408507989986013934690 8/)
2 =(/	0.30353099910334311154769579070 9,	0.93417235896271569645111862354 8,	0.18759247408507989986013934690 8/)
3 =(/	-0.79465447229176612295553092832 7,	0.57735026918962576450914878050 3,	0.18759247408507989986013934690 8/)
4 =(/	-0.79465447229176612295553092832 7,	-0.57735026918962576450914878050 2,	0.18759247408507989986013934690 8/)
5 =(/	0.30353099910334311154769579070 9,	-0.93417235896271569645111862354 8,	0.18759247408507989986013934690 8/)
6 =(/	0.79465447229176612295553092832 7,	0.57735026918962576450914878050 2,	-0.18759247408507989986013934690 8/)
7 =(/	-0.30353099910334311154769579070 9,	0.93417235896271569645111862354 8,	-0.18759247408507989986013934690 8/)
8 =(/	-0.98224694637684602281567027523 5,	0.000000000000000000000000000000,	-0.18759247408507989986013934690 8/)
9 =(/	-0.30353099910334311154769579070 9,	-0.93417235896271569645111862354 8,	-0.18759247408507989986013934690 8/)
10 =(/	0.79465447229176612295553092832 7,	-0.57735026918962576450914878050 3,	-0.18759247408507989986013934690 8/)
11 =(/	0.49112347318842301140783513761 7,	0.35682208977308993194196984304 6,	-0.79465447229176612295553092832 8/)
12 =(/	-0.18759247408507989986013934690 7,	0.57735026918962576450914878050 2,	-0.79465447229176612295553092832 8/)
13 =(/	-0.60706199820668622309539158142 0,	0.000000000000000000000000000000,	-0.79465447229176612295553092832 8/)
14 =(/	-0.18759247408507989986013934690 7,	-0.57735026918962576450914878050 2,	-0.79465447229176612295553092832 8/)
15 =(/	0.49112347318842301140783513761 7,	-0.35682208977308993194196984304 6,	-0.79465447229176612295553092832 8/)
16*=(/	0.60706199820668622309539158142 0,	0.000000000000000000000000000000,	0.79465447229176612295553092832 8/)
17 =(/	0.18759247408507989986013934690 7,	0.57735026918962576450914878050 2,	0.79465447229176612295553092832 8/)
18 =(/	-0.49112347318842301140783513761 7,	0.35682208977308993194196984304 6,	0.79465447229176612295553092832 8/)
19 =(/	-0.49112347318842301140783513761 7,	-0.35682208977308993194196984304 6,	0.79465447229176612295553092832 8/)
20 =(/	0.18759247408507989986013934690 7,	-0.57735026918962576450914878050 2,	0.79465447229176612295553092832 8/)

Face normals:	X	Y	Z
1 =(/	0.71076056746744485830848048142 5,	0.000000000000000000000000000000,	-0.35538028373372242915424024071 3/)
2 =(/	0.21963709427902184690064534380 8,	0.67597346921555457077250093023 0,	-0.35538028373372242915424024071 3/)
3 =(/	-0.57501737801274427605488558452 0,	0.41777457946839344509388323691 1,	-0.35538028373372242915424024071 3/)
4 =(/	-0.57501737801274427605488558452 0,	-0.41777457946839344509388323691 1,	-0.35538028373372242915424024071 3/)
5 =(/	0.21963709427902184690064534380 7,	-0.67597346921555457077250093023 0,	-0.35538028373372242915424024071 3/)
6 =(/	0.57501737801274427605488558452 0,	0.41777457946839344509388323691 1,	0.35538028373372242915424024071 3/)
7 =(/	-0.21963709427902184690064534380 8,	0.67597346921555457077250093023 0,	0.35538028373372242915424024071 3/)
8 =(/	-0.71076056746744485830848048142 5,	0.000000000000000000000000000000,	0.35538028373372242915424024071 3/)
9 =(/	-0.21963709427902184690064534380 8,	-0.67597346921555457077250093023 0,	0.35538028373372242915424024071 3/)
10 =(/	0.57501737801274427605488558452 0,	-0.41777457946839344509388323691 1,	0.35538028373372242915424024071 3/)
11 =(/	0.000000000000000000000000000000,	0.000000000000000000000000000000,	-0.79465447229176612295553092832 8/)
12 =(/	-0.000000000000000000000000000000,	0.000000000000000000000000000000,	0.79465447229176612295553092832 8/)

Web Link to this document: http://ftp.setterholm.com/Fortran/Dodec.pdf

Because no one is perfect, and *lawyers* abound: recognize that this is simply “*my best shot*… as of today”.

```
Subroutine DodecModel16(iP)
!2013.10.15.2200cdt JMS- Dodecahedron - the 12-face Platonic Solid
!                        which fits inside a sphere of radius=1.0
!                      - Traveler2/Athlon64/WinXPPro/APF9.0/Ogl1.2.1
!------                - Reference: Wikipedia's article on "Dodecahedron".
 implicit none
 !-----                                                          Arguments
 integer*4::iP
 !-----                                                          Internals
!type::  DodecRec  ;sequence
  integer*4::Init
  !Theoretical value:
  real*16  ::EdgeL          !Edge length   =(sqrt(5.)-1.)/sqrt(3.)
  !My model parameters:
  real*16  :: v1Ang         !Vertex -  #1- Pitch Angle w.r.t. the X-Y plane.
  real*16  ::v16Ang         !       - #16-
  !Indices:
  integer*4::iFace(5,12)    !Face   - vertex  # list (CCW looking out)
  integer*4::iEdge(2,30)    !Edge   - vertex  # list
  integer*4::iVertex(3,20)  !vertex - vNormal # list (CCW looking out)
  !Computed Vertices & Normals
  !Note: I use homogeneous vectors, but you can ignore the 4th component here.
  real*16  ::Vertex(4,20)   !   Vertex   vector                 (X,Y,Z,1.)
  real*16  ::vNormal(4,12)  !Face normal vector                 (X,Y,Z,1.)
  real*16  ::EdgeLength(30) !Edge- lengths- ~= EdgeL ...Yes!
  real*16  ::EdgeLErrorRms  !             - RMS error
!end type DodecRec ;type(DodecRec)::Dd16
  !-- Internal use variables
  integer*4::i,iDel,iDel2,j
  real*16  ::v0=0._16  !0.
  real*16  ::v1=1._16  !1.
  real*16  ::Angle     !Yaw angle-
  real*16  ::Ver1(4)   !Vertex vector- # 1- used as a template  (X,Y,Z,1.)
  real*16  ::Ver16(4)  !             - #16-                    (X,Y,Z,1.)
  real*16  ::sY,cY     !Yaw  - cosine, sine
  real*16  ::sP,cP     !Pitch- cosine, sine
  real*16  ::vTemph(4) !Temporary vector                        (X,Y,Z,1.)

   !----- Dodecahedron initilization - Jeff Setterholm's algorithm:
   if(Init.eq.0) then
     !---
     EdgeL  =(sqrt(5._16)-1._16)/sqrt(3._16)       !Theoretical value.
           != 0.71364417954617986383839396860 92_16
     V1Ang =-10.81231696357170629128494477447230_16 !=Pitch angle  v#1
     V16Ang=-52.62263185935030435714286150500763_16 !=Pitch angle v#16

    !-- Index faces & edges:
     iDel =5 ;iDel2=0
     do i=1,5 !Index faces(10 of 12 sets) {-CCW looking out} & all edges(30):
         iFace(1:5,i    )=(/i  ,i+5,i+10,i+ 9+iDel ,i+4+iDel /) !Face[ 1, 5]
         iFace(1:5,i  +5)=(/i+5,i  ,i+15,i+16-iDel2,i+1-iDel2/) !Face[ 6,10]
         iEdge(1:2,i*6-5)=  iFace(1:2,i)                    !Edge(1, 7,13,19,25)
         iEdge(1:2,i*6-4)=  iFace(2:3,i)                    !Edge(2, 8,14,20,26)
         iEdge(1:2,i*6-3)=  iFace(3:4,i)                    !Edge(3, 9,15,21,27)
         iEdge(1:2,i*6-2)=(/iFace(1,i+5),iFace(5,i+5)/)     !Edge(4,10,16,22,28)
         iEdge(1:2,i*6-1)=(/iFace(5,i+5),iFace(4,i+5)/)     !Edge(5,11,17,23,29)
         iEdge(1:2,i*6  )=(/iFace(4,i+5),iFace(3,i+5)/)     !Edge(6,12,18,24,30)
       iVertex(1:3,i    )=(/i  ,i+4+iDel ,i+5/)             !iVertex[ 1, 5]
       iVertex(1:3,i  +5)=(/i+5,i+1-iDel2,i  /)             !iVertex[ 6,10]
       iVertex(1:3,i +10)=(/i  ,i+1-iDel2, 11/)             !iVertex[11,15]
       iVertex(1:3,i +15)=(/i+5,i+4+iDel , 12/)             !iVertex[16,20]
       if(i.eq.1) iDel =0
       if(i.eq.4) iDel2=5
     enddo!i
         iFace(  1:5,11)=(/13,14,15,11,12/)                   !Face #11-Top
         iFace(  1:5,12)=(/19,18,17,16,20/)                   !Face #12-Bottom
     Init=1
   endif !Init=0
```

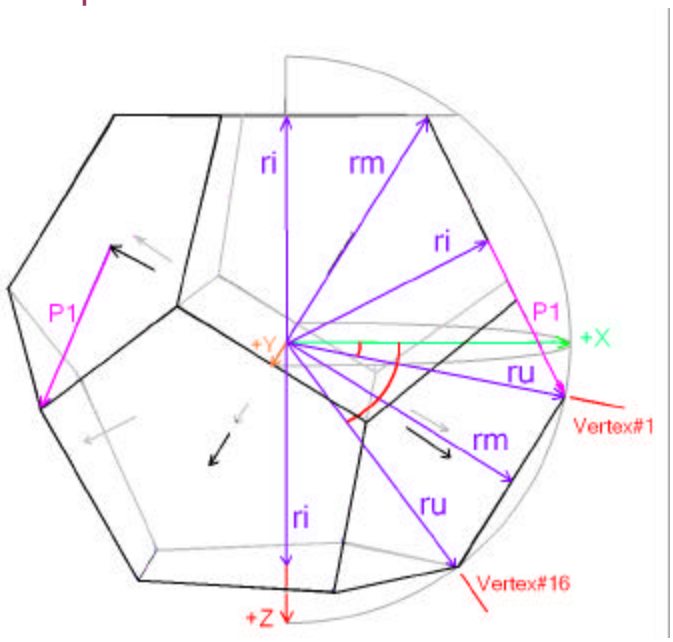
```
    !----- From here on: V1Ang & V16Ang quantify the model...
    !-- Setup the Vertex templates:
    call SinCos16(v1Ang ,sP,cP) ;Ver1 =(/cP,v0,-sP,v1/) != vertex#1
    call SinCos16(v16Ang,sP,cP) ;Ver16=(/cP,v0,-sP,v1/) != vertex#16
    do i=1,5  !-- Yaw the two vertex templates to define all the vertices:
      Angle=(i-1)*72._16       ;call SinCos16(Angle,sY,cY)
       Vertex( 1:4,i  )=(/ Ver1(1)*cY, Ver1(1)*sY,  Ver1(3),v1/)!Ver[ 1, 5]
       Vertex(1:4,i+15)=(/Ver16(1)*cY,Ver16(1)*sY, Ver16(3),v1/)!Ver[16,20]
      Angle=(i-1)*72._16+36._16 ;call SinCos16(Angle,sY,cY)
       Vertex( 1:4,i+5)=(/ Ver1(1)*cY, Ver1(1)*sY, -Ver1(3),v1/)!Ver[ 6,10]
       Vertex(1:4,i+10)=(/Ver16(1)*cY,Ver16(1)*sY,-Ver16(3),v1/)!Ver[11,15]
    enddo!i

    !-- Compute each Face Normal vector by averaging its five vertices:
    do i=1,12 ;vNormal(1:4,i)=0._16
      do j=1,5 ;vNormal(1:4,i)=vNormal(1:4,i)+Vertex(1:4,iFace(j,i));enddo!j
      vNormal(1:4,i)=vNormal(1:4,i)/5._16
    enddo!i

    !-- Compute the resulting Edge Lengths:
    EdgeLErrorRms=0._16
    do i=1,30
      vTemph=Vertex(1:4,iEdge(2,i))-Vertex(1:4,iEdge(1,i))
      EdgeLength(i)= &
                  vTemph(1)*vTemph(1)+vTemph(2)*vTemph(2)+vTemph(3)*vTemph(3)
      call Sqrt16(EdgeLength(i),EdgeLength(i),0)
      EdgeLErrorRms=EdgeLErrorRms+(EdgeLength(i)-EdgeL)*(EdgeLength(i)-EdgeL)
    enddo!i
    EdgeLErrorRms=EdgeLErrorRms/30._16
    call Sqrt16(EdgeLErrorRms,EdgeLErrorRms,0)

    !----- Report the results:
    if(iP.gt.5) then
      write(iP,"(f36.32,' = V#1  pitch angle')") v1Ang
      write(iP,"(f36.32,' = V#16 pitch angle')") v16Ang
      write(iP,"(/'Vertices:')")
      do i=1,20
        write(iP,"(i2,' =(/',2(f34.30,','),f34.30,'/)')") i,Vertex( 1:3,i)
      enddo!i
      write(iP,"(/'Face normals:')")
      do i=1,12
        write(iP,"(i2,' =(/',2(f34.30,','),f34.30,'/)')") i,vNormal( 1:3,i)
      enddo!i
      !-- Indices & Lengths:
      write(iP,"(/'       CCW looking out    CCW looking out' )")
      write(iP,"( 'Face:    VertexIndex   Vertex:  FaceIndex '\)")
      write(iP,"( 'Edge:VertexIndex & ComputedLength')")
      do i=1,30  ! ...print in three side-by-side columns…
        if(i.le.12) then
          write(iP,"(' #',i2,'=( ',4(i2,','),i2,'/)'\)") i,iFace(1:5,i)
         else; write(iP,"('                            '\)")
        endif !i<=12
        if(i.le.20) then
          write(iP,"('   #',i2,'=( ',2(i2,','),i2,'/)'\)") i,iVertex(1:3,i)
         else; write(iP,"('                        '\)")
        endif !i<=20
        write(iP,"('   #',i2,'=( ',1(i2,','),i2,'/)'\)") i,iEdge(1:2,i)
        write(iP,"(' :',f35.32)") EdgeLength(i)
      enddo!i
        write(iP,"(/36x,'Theoretical Edge Length=',f35.32)") EdgeL
        write(iP,"( 72x,'1         2         3')")
        write(iP,"( 63x,'12345678901234567890123456789012')")
        write(iP,"( 67x,'RMS Error=',d13.6,' : ^')") EdgeLErrorRms
    endif !iP>5
    return
End Subroutine DodecModel16
```

Relating my model to Wikipedia's *exact* model:



Define the radii of three spheres:

ru = the circumscribing (outside) sphere touching all 20 vertices:
= (1 + sqrt(5)) *sqrt(3) /4
@ 1.401258538444073544676677935322O7

rm = the sphere tangent to the middle of all 30 edges:
= (sqrt(5) +3)/4
@ 1.309016994374947424102293417182S2

ri = the sphere tangent to the center of all 12 faces:
= (sqrt(sqrt(5)*11/10+5/2))/2
@ 1.113516364411606735194375039486S6

P1 = radius of the circle touching the five vertices of each face"
= sqrt(ru*ru-ri*ri)
@ 0.850650808352039932181540497063O2

The theoretical value of my model parameters become:

sinV1Ang = (ri-2*P1)/(ru*sqrt(5))
@ -0.187592474085079899860139346907G1
V1Ang = arcsine(sinV1Ang)

V1Ang @ -10.812316963571706291284944774473S3 degrees

-and-

cosV16Ang = P1 / ru
@ 0.607061998206686223095391581419S9
V16Ang = - arcsine(cosV16Ang)
V16Ang @ -52.622631859350304357142861505100S2 degrees

Wikipedia Model:

Phi =(1+sqrt(5))/2 *exactly* = the "Golden Ratio" @ 1.61803398874989484820458683436564

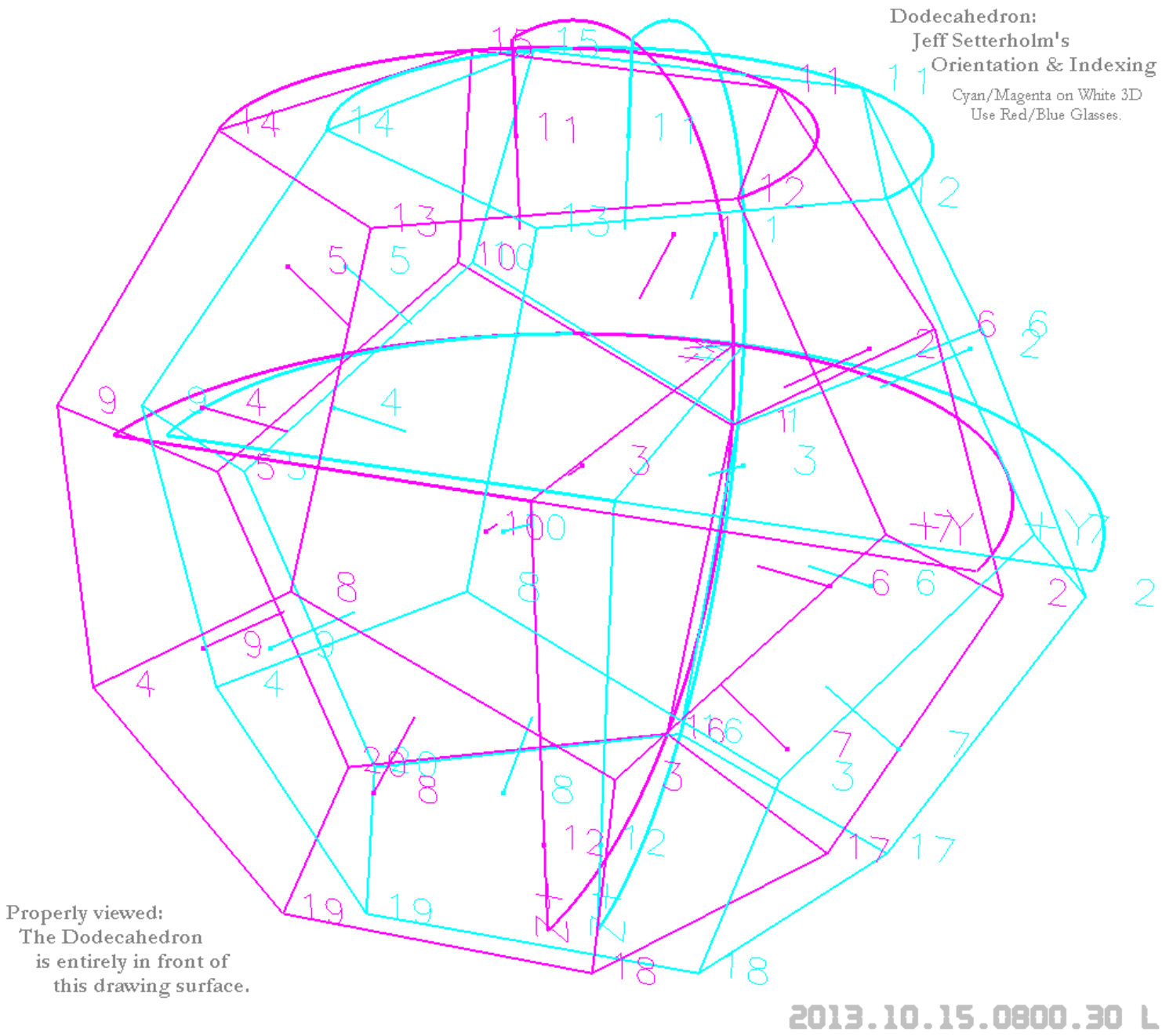
dPhi = 1 / Phi @ 0.61803398874989484820458683436564

W#: S#: Numeric values: :vertex algebraic summary

```
  1 = 14 =(/  -1  ,  -1  ,  -1  /)
  2 =  4 =(/  -1  ,  -1  ,   1  /)
  3 = 12 =(/  -1  ,   1  ,  -1  /)
  4 =  3 =(/  -1  ,   1  ,   1  /)
  5 = 10 =(/   1  ,  -1  ,  -1  /)
  6 = 20 =(/   1  ,  -1  ,   1  /)
  7 =  6 =(/   1  ,   1  ,  -1  /)
  8 = 17 =(/   1  ,   1  ,   1  /)
  9 = 15 =(/   0  , -dPhi,  -phi/)
 10 = 19 =(/   0  , -dPhi,   phi/)
 11 = 11 =(/   0  ,  dPhi,  -phi/)
 12 = 18 =(/   0  ,  dPhi,   phi/)
 13 =  9 =(/ -dPhi,  -Phi,   0  /)
 14 =  7 =(/ -dPhi,   Phi,   0  /)
 15 =  5 =(/  dPhi,  -Phi,   0  /)
 16 =  2 =(/  dPhi,   Phi,   0  /)
 17 = 13 =(/  -Phi,   0  , -dPhi/)
 18 =  8 =(/  -Phi,   0  ,  dPhi/)  ...scale and rotate the model:
 19 =  1 =(/   Phi,   0  , -dPhi/)  Divide by sqrt(3) @  1.732050807568877293527446341505872
 20 = 16 =(/   Phi,   0  ,  dPhi/)  and pitch down by (90-arccosine(rm/ru)-arccosine(ri/ru)) degrees
                                                    @ -31.717474411461005324213903139828932 degrees.
```

W#: ... the resulting Wiki-Vertices in “Setterholm Model” order are:

```
19 =(/  0.98224694637684602281567027523490,  0.00000000000000000000000000000000,  0.18759247408507989986013934690910/)
16 =(/  0.30353099910334311154769579070970,  0.93417235896271569645111862354800,  0.18759247408507989986013934690810/)
 4 =(/ -0.79465447229176612295553092832790,  0.57735026918962576450914878050200,  0.18759247408507989986013934690640/)
 2 =(/ -0.79465447229176612295553092832790, -0.57735026918962576450914878050200,  0.18759247408507989986013934690640/)
15 =(/  0.30353099910334311154769579070970, -0.93417235896271569645111862354800,  0.18759247408507989986013934690810/)
 7 =(/  0.79465447229176612295553092832790,  0.57735026918962576450914878050200, -0.18759247408507989986013934690640/)
14 =(/ -0.30353099910334311154769579070970,  0.93417235896271569645111862354800, -0.18759247408507989986013934690810/)
18 =(/ -0.98224694637684602281567027523490,  0.00000000000000000000000000000000, -0.18759247408507989986013934690910/)
13 =(/ -0.30353099910334311154769579070970, -0.93417235896271569645111862354800, -0.18759247408507989986013934690810/)
 5 =(/  0.79465447229176612295553092832790, -0.57735026918962576450914878050200, -0.18759247408507989986013934690640/)
11 =(/  0.49112347318842301140783513761880,  0.35682208977308993194196984304610, -0.79465447229176612295553092832680/)
 3 =(/ -0.18759247408507989986013934690640,  0.57735026918962576450914878050200, -0.79465447229176612295553092832790/)
17 =(/ -0.60706199820668622309539158141880,  0.00000000000000000000000000000000, -0.79465447229176612295553092832850/)
 1 =(/ -0.18759247408507989986013934690640, -0.57735026918962576450914878050200, -0.79465447229176612295553092832790/)
 9 =(/  0.49112347318842301140783513761880, -0.35682208977308993194196984304610, -0.79465447229176612295553092832680/)
20 =(/  0.60706199820668622309539158141880,  0.00000000000000000000000000000000,  0.79465447229176612295553092832850/)
 8 =(/  0.18759247408507989986013934690640,  0.57735026918962576450914878050200,  0.79465447229176612295553092832790/)
12 =(/ -0.49112347318842301140783513761880,  0.35682208977308993194196984304610,  0.79465447229176612295553092832680/)
10 =(/ -0.49112347318842301140783513761880, -0.35682208977308993194196984304610,  0.79465447229176612295553092832680/)
 6 =(/  0.18759247408507989986013934690640, -0.57735026918962576450914878050200,  0.79465447229176612295553092832790/)
```


Dodecahedron:
Jeff Setterholm's
Orientation & Indexing
Cyan/Magenta on White 3D
Use Red/Blue Glasses.
Properly viewed:
The Dodecahedron
is entirely in front of
this drawing surface.
2013.10.15.0800.30 L