Gernot Hoffmann

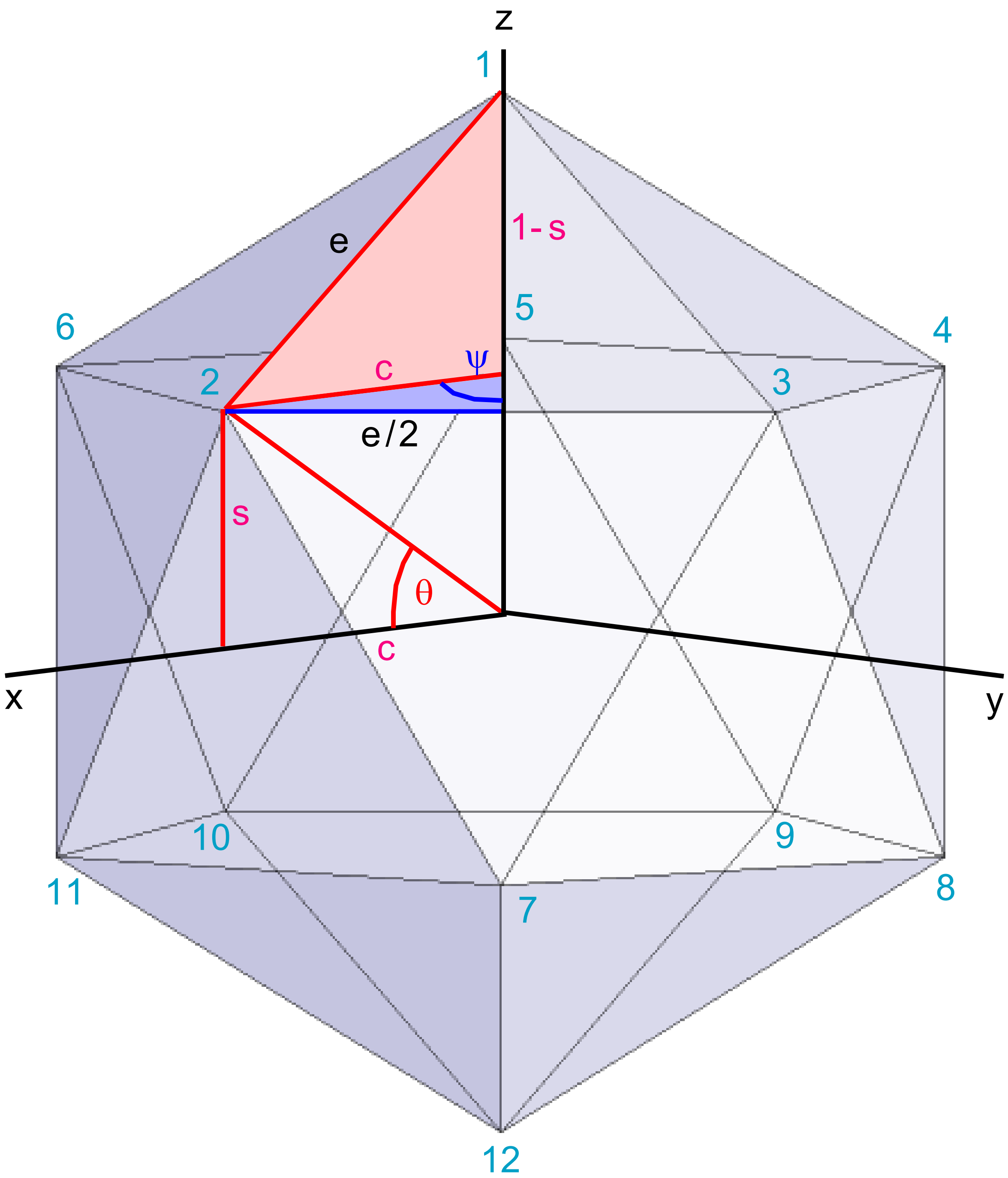
Sphere Tessellation by Icosahedron Subdivision

Contents

Best View Acrobat 5 / Zoom 100%
Smooth line art / Smooth text,
Do not smooth images

1. Vertex Coordinates

An Icosahedron has 12 vertices, 30 edges and 20 triangles. The vertices 1 and 12 are at the poles, the vertices 2 to 11 are on two pentagons. The only unknown angle is the latitude θ. As a result of some basic geometry we find $\theta = 26.565°$. The vertices are easily found in sphere coordinates.




```
dps:=2*pi/5;
sps:=2*Sqr(sic(dps/2));
Quadglei(sps,-1,1-sps,r1,r2,i1,i2,flag);
sth:=r2; { smaller root     sin(theta)  }
cth:=Sqrt(1-Sqr(sth)); {    cos(theta)  }
psi:=0;
For k:=2 to 6 Do
 Begin
 With xr[k] Do
  Begin
  x:=cth*coc(psi); y:=cth*sic(psi); z:=sth;
  End;
 psi:=psi+dps;
 End;
sth:=-sth;
psi:=0.5*dps;
For k:=7 to 11 Do
 Begin
 With xr[k] Do
  Begin
  x:=cth*coc(psi); y:=cth*sic(psi); z:=sth;
  End;
 psi:=psi+dps;
 End;
With xr[ 1] Do Begin x:=0; y:=0; z:=+1; End;
With xr[12] Do Begin x:=0; y:=0; z:=-1; End;
```

$r = 1$ radius

e edge length

$\psi = 2\pi/10$

$c = \cos\theta$

$s = \sin\theta$

$e/2 = \cos\theta \sin\psi$

$$e^2 = 4\cos^2\theta \sin^2\psi$$
$$= \cos^2\theta + (1-\sin\theta)^2$$

Quadratic equation for s

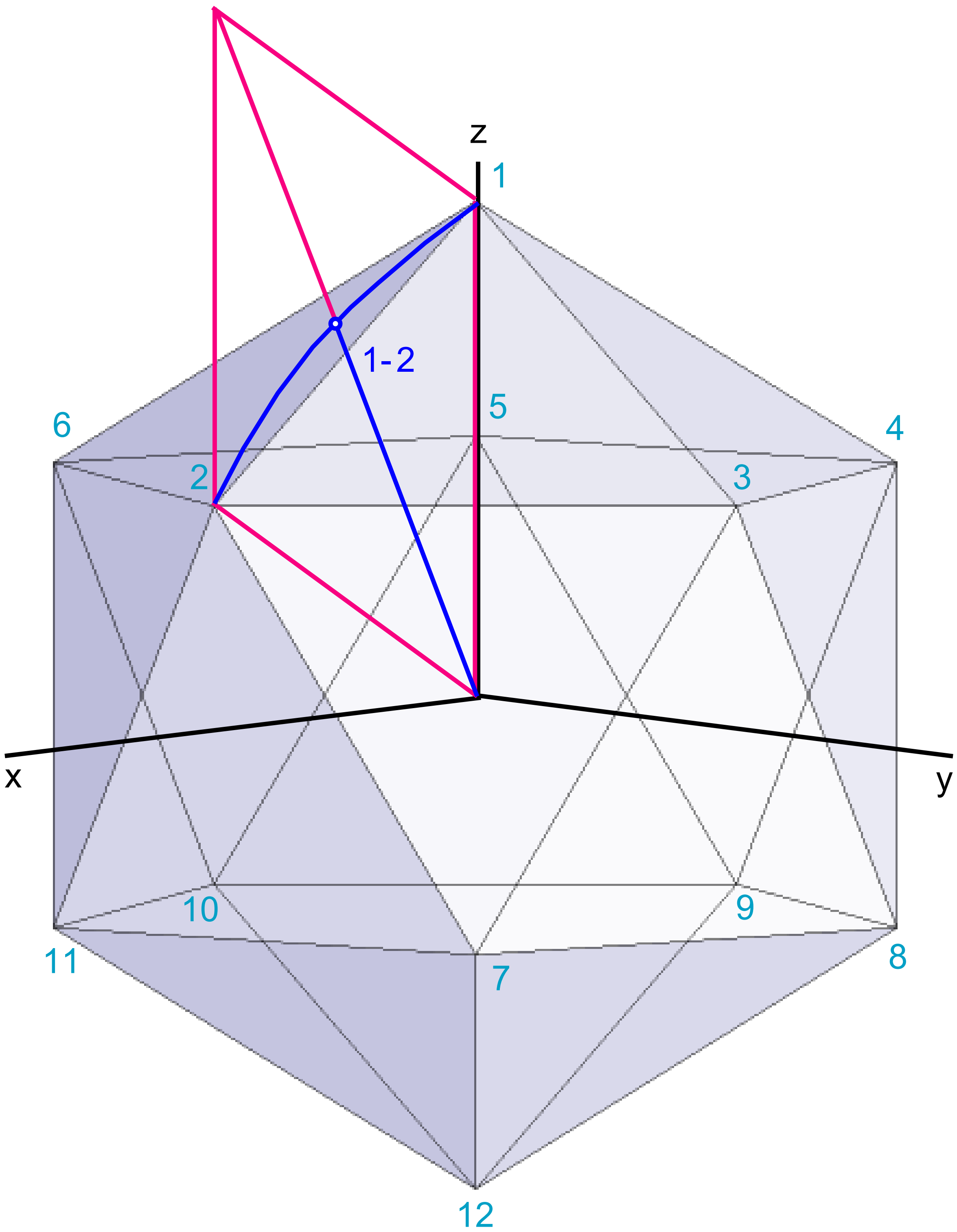
$$2\sin^2\psi\, s^2 - s + 1 - 2\sin^2\psi = 0$$

Smaller root delivers

$\theta = 26.565^{\circ}$

2. Edge Subdivision

An edge is divided by the normalized sum of two vertex vectors, new point on the sphere with radius 1.

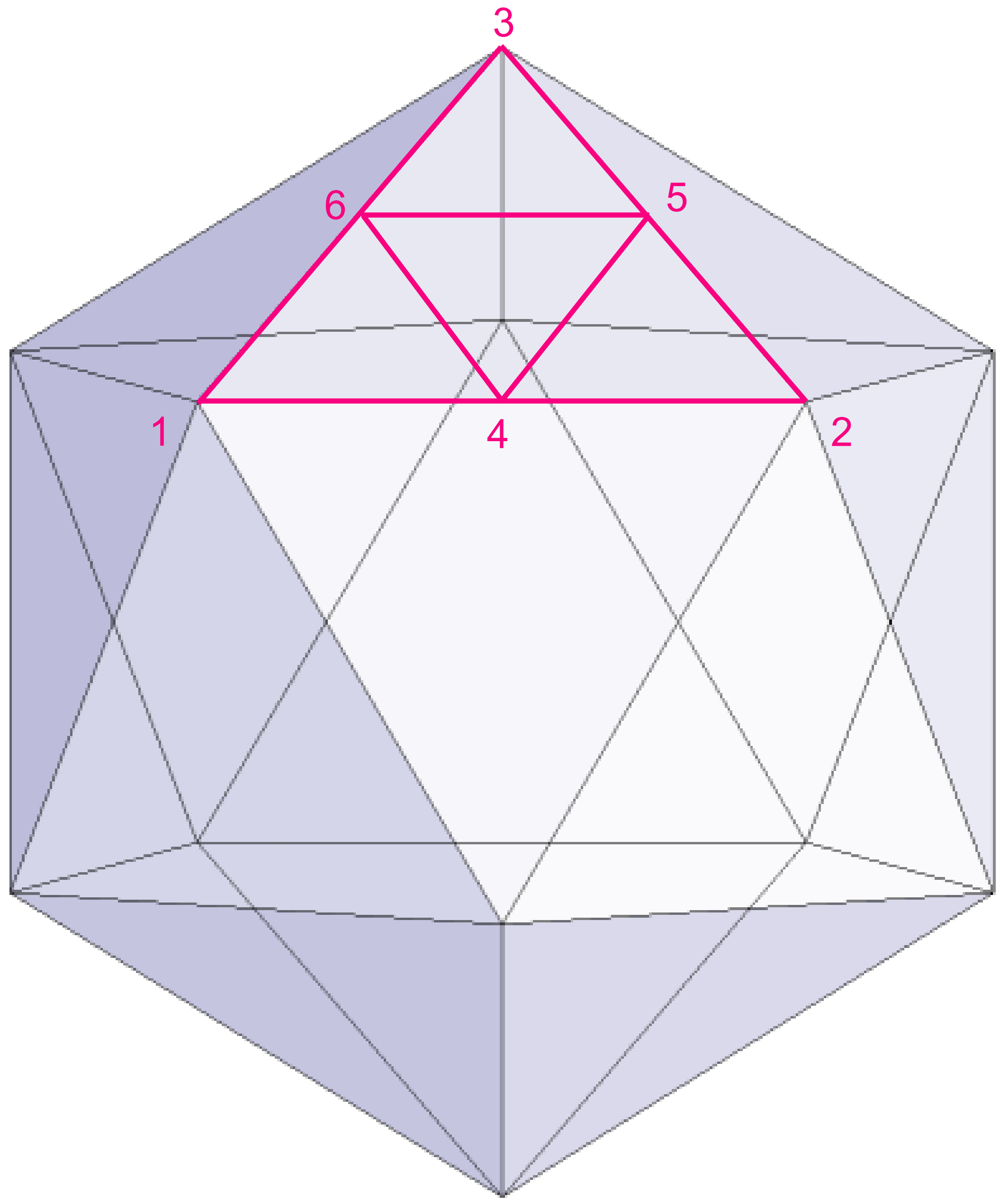



```
Procedure SubDivi(x1,x2: XYZ; Var x3: XYZ);
Var r: Single;
Begin
With x3 Do
 Begin
  x:=x1.x+x2.x; y:=x1.y+x2.y; z:=x1.z+x2.z;
  r:=1/RootSqr(x,y,z);
  x:=r*x; y:=r*y; z:=r*z;
 End;
End;
```

```
Some important variables

XYZ is a coordinate record          x,y,z
PQE is a pixel and z-buffer record p,q,e
PC  is a color record

Type    Xtr=Record
        t1,t2,t3: XYZ; End;
Type    Ptr=Record
        pa1,pb1,pa2,pb2,pa3,pb3: Single;
End;
Const   ntr=20*4*4*4;
Var     lev,lex,tri: Integer;
        Ra,hue     : Single;
        x1,x2,x3,n : XYZ;
        p1,p2,p3   : PQE;
        c0,c1,c2,c3: PC;
        Pbody,Psoft,Pgrid: Integer;
        xr       : Array[1.. 12] Of XYZ;
        Lt1,Lt2  : Array[1..ntr] Of ^Xtr;
        Par      : Array[1..ntr] Of ^Ptr;
```

3. Triangle Subdivision

A triangle 1-2-3 delivers by edge division four triangles 1-4-6, 4-2-5, 6-5-3 and 4-5-6.
The vertex numbers are local in this example. Two triangle lists are used, Lt1 and Lt2.
Set 1-2-3 is stored in Lt1, then copied to Lt2.
The subdivided set is stored in Lt1. This is much simpler than true recursive programming.

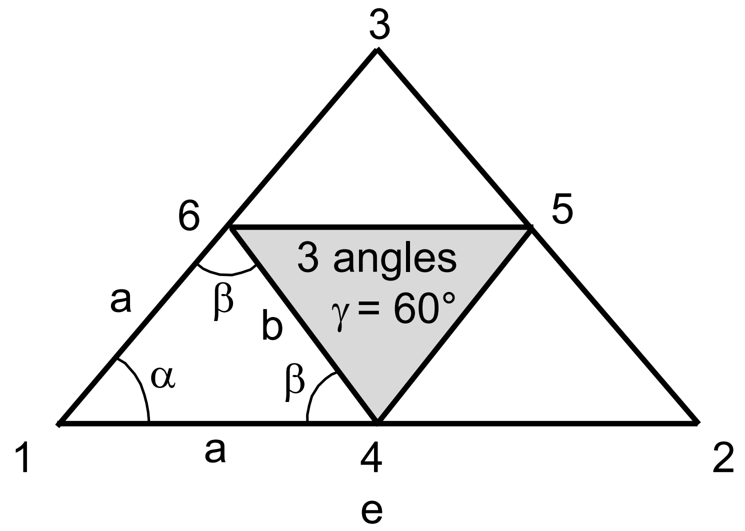



```
Procedure IcoRecu;
{ Recursive Subdivision
  Old 1-2-3
  New 1-4-6, 4-5-6, 4-2-5, 6-5-3
          3
      6     5
    1    4    2       }
Var x1,x2,x3,x4,x5,x6: XYZ;
    k: Integer;
Begin
For i:=1 to tri Do Lt2[i]^:=Lt1[i]^;
k:=0;
For i:=1 to tri Do
Begin
With Lt2[i]^Do
 Begin
 x1:=t1; x2:=t2; x3:=t3;
 End;
 SubDivi(x1,x2,x4);
 SubDivi(x2,x3,x5);
 SubDivi(x3,x1,x6);
```

```
With Lt1[k]^ Do
  Begin t1:=x1; t2:=x4; t3:=x6;
  End;
 Inc(k);
 With Lt1[k]^ Do
  Begin t1:=x4; t2:=x5; t3:=x6;
  End;
 Inc(k);
 With Lt1[k]^ Do
  Begin t1:=x4; t2:=x2; t3:=x5;
  End;
 Inc(k);
 With Lt1[k]^ Do
  Begin t1:=x6; t2:=x5; t3:=x3;
  End;
 End;
 tri:=k;
End;
```

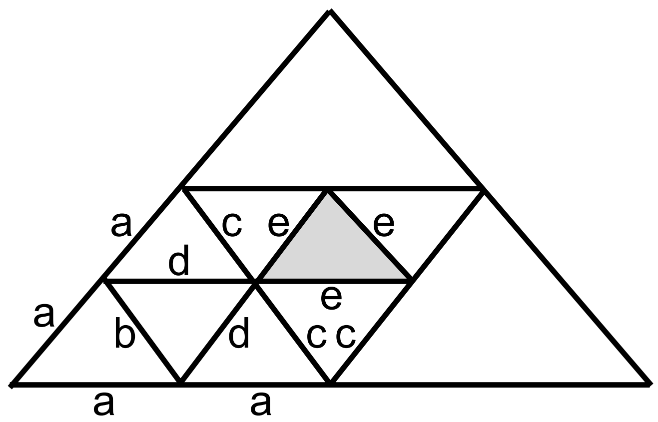
4. Edge Lengths

An original triangle 1-2-3 (subdivision level 0) delivers by subdivision four triangles 1-4-6,4-2-5, 6-5-3 and 4-5-6. This is subdivision level 1. The vertex numbers are local in this example.



Subdivision level 0 , 20 triangles:
e = 1.051462
α = 60°

Subdivision level 1 , 80 triangels:
a = 0.546533
b = 0.618034
α = 68.862°
β = 55.569°
γ = 60.000°

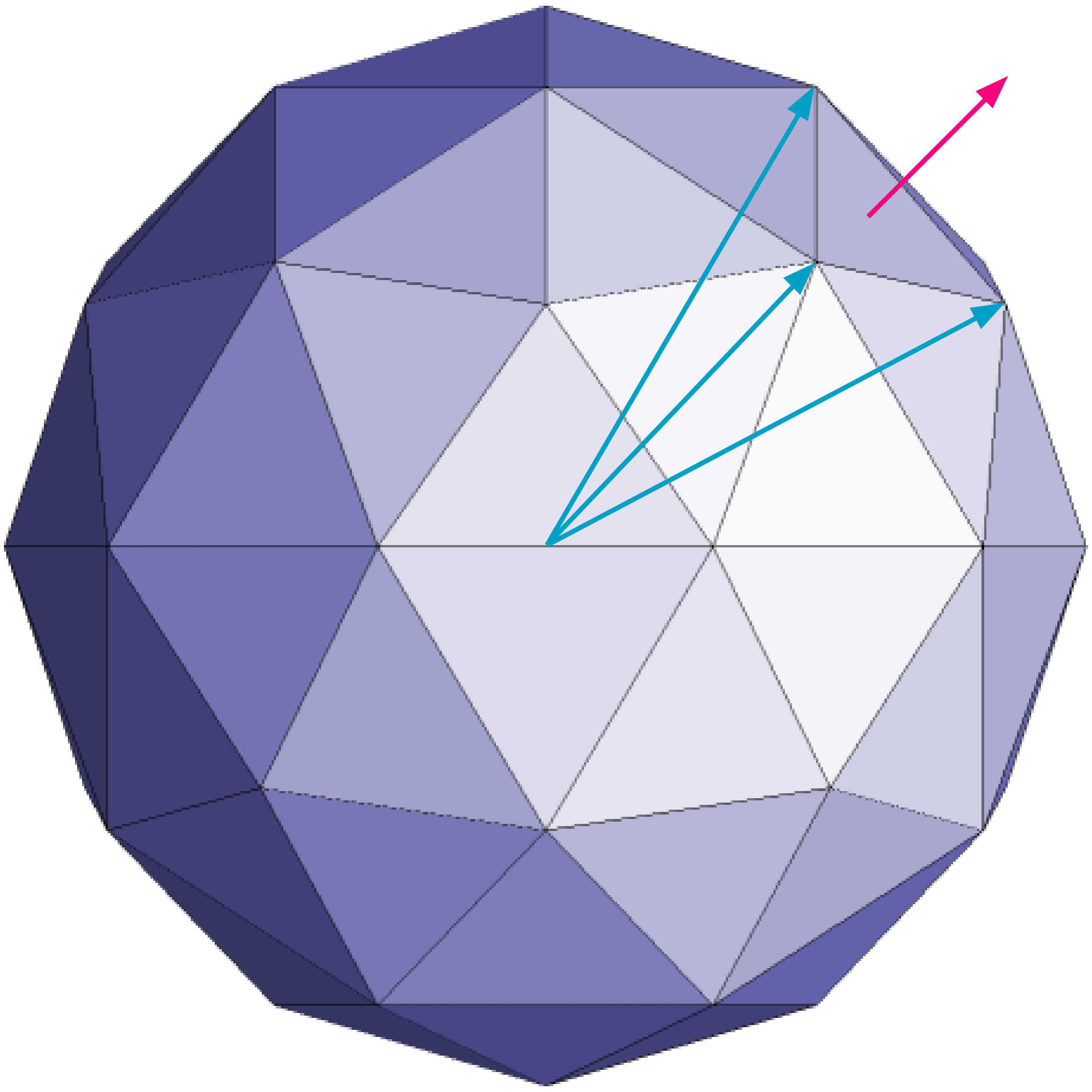


Subdivision level 2 , 320 triangels:
a = 0.275905
b = 0.321244
c = 0.312869
d = 0.285473
e = 0.324920
α = 54.397° ... 71.206° (8 angles)

For architecture it would be interesting to use the same edgelengths with a minor deviation from the true sphere shape. This is not possible. Equal edgelenghts can be achieved only by a flat subdivision of the original triangles. Any projection towards the sphere surface creates different edgelengths.

5. Normal Vectors

Because the Icosahedron is embedded into a sphere, each vertex vector is also the normal vector at this position. For Gouraud shading the normal vector is assigned directly to the vertex. For facetted shading it is necessary to take the mean value of three vertices for the triangle.



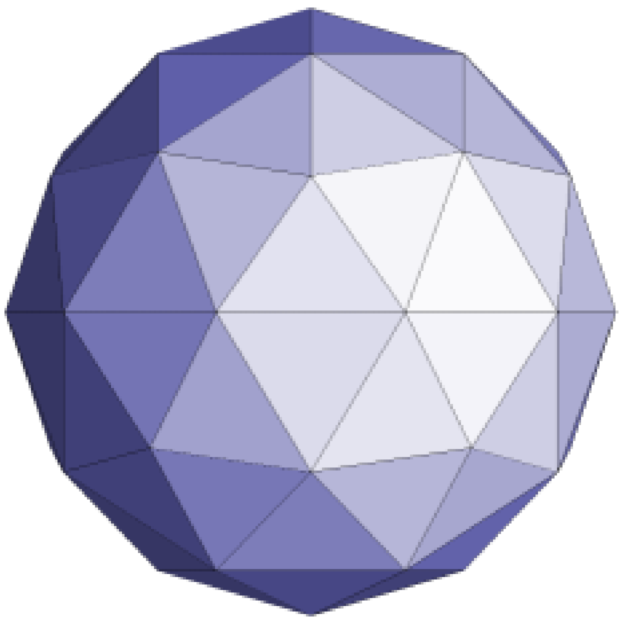
```
Procedure IcoShow;
Var i,sel: Integer;
Begin
 sel:=1;
 For i:=1 to tri Do
  Begin
  With Lt1[i]^ Do
    Begin
    x1:=t1; x2:=t2; x3:=t3;
    End;
 x1.x:=Ra*x1.x; x1.y:=Ra*x1.y; x1.z:=Ra*x1.z;
 x2.x:=Ra*x2.x; x2.y:=Ra*x2.y; x2.z:=Ra*x2.z;
 x3.x:=Ra*x3.x; x3.y:=Ra*x3.y; x3.z:=Ra*x3.z;
    ObjTra3D(x1,x1);        { Rotate object }
    ObjTra3D(x2,x2);
    ObjTra3D(x3,x3);
    Abbild3R(x1,p1,sel);    { Map to Raster }
    Abbild3R(x2,p2,sel);
    Abbild3R(x3,p3,sel);
    If Psoft=1 Then
    Begin                   { Gouraud       }
    LicMod(x1,x1,c1,sel);   { Luminance     }
    LicMod(x2,x2,c2,sel);
    LicMod(x3,x3,c3,sel);
    End Else
```

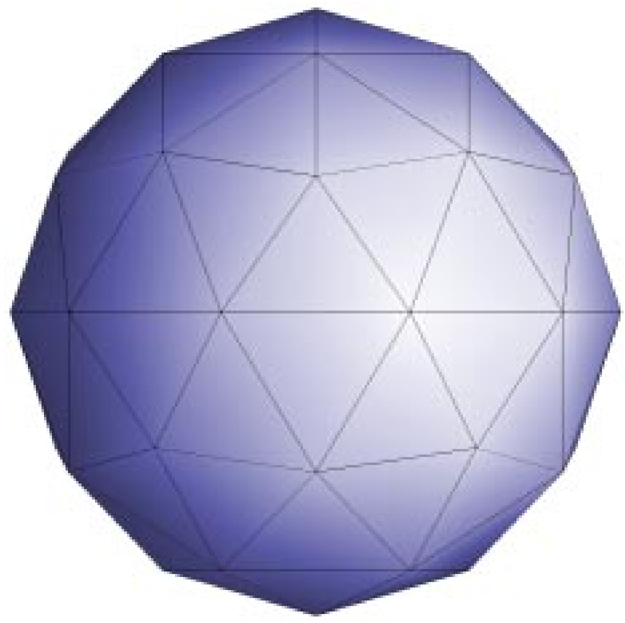
```
    Begin                   { Facetted  }
    n.x:=x1.x+x2.x+x3.x;
    n.y:=x1.y+x2.y+x3.y;
    n.z:=x1.z+x2.z+x3.z;
    LicMod(x1,n,c1,sel);
    c2:=c1; c3:=c1;
    End;
    If Pbody=1 Then
    FillTriS(p1,p2,p3,c1,c2,c3,sel);
    If Pgrid=1 Then
    DrawSTria(p1,p2,p3,c0,c0,c0,sel);
  End;
End;
```

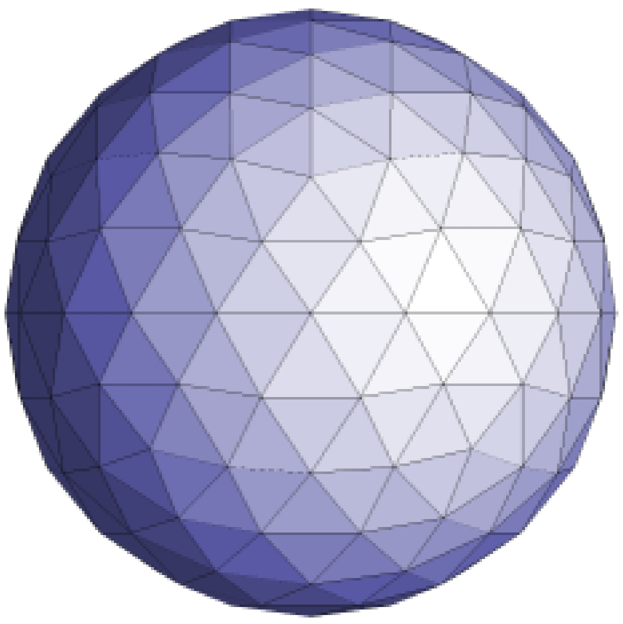
6. Subdivided Icosahedrons

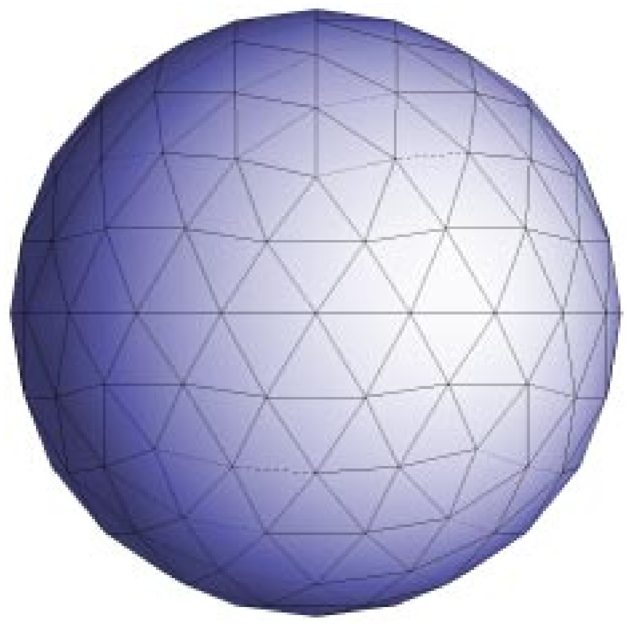
Left side facetted shading, right side Gouraud shading. Grids are Z-buffered

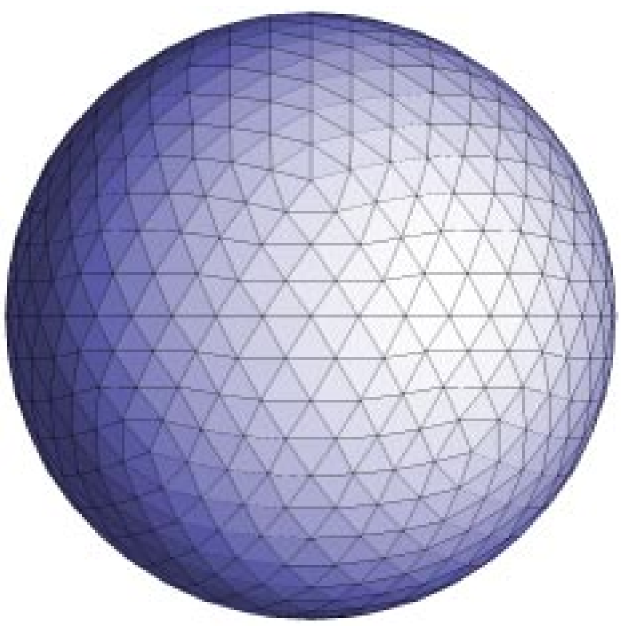
Level 1	80 triangles
Level 2	320 triangles
Level 3	1280 triangles
Level 4	5120 triangles (p.9)

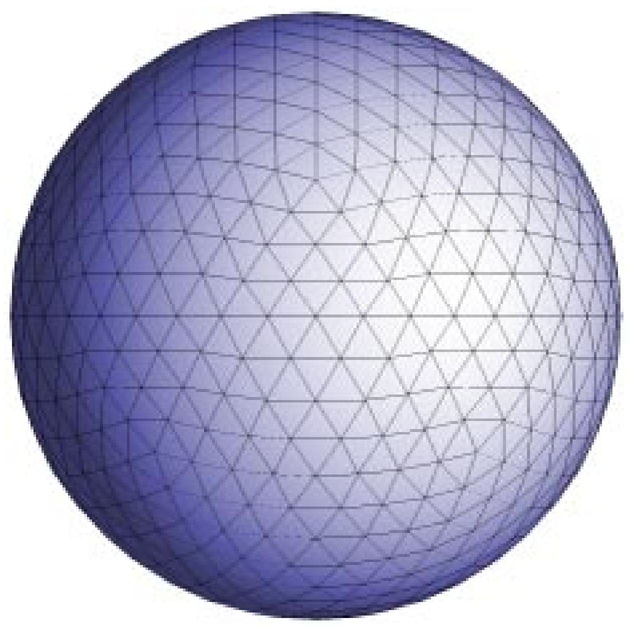






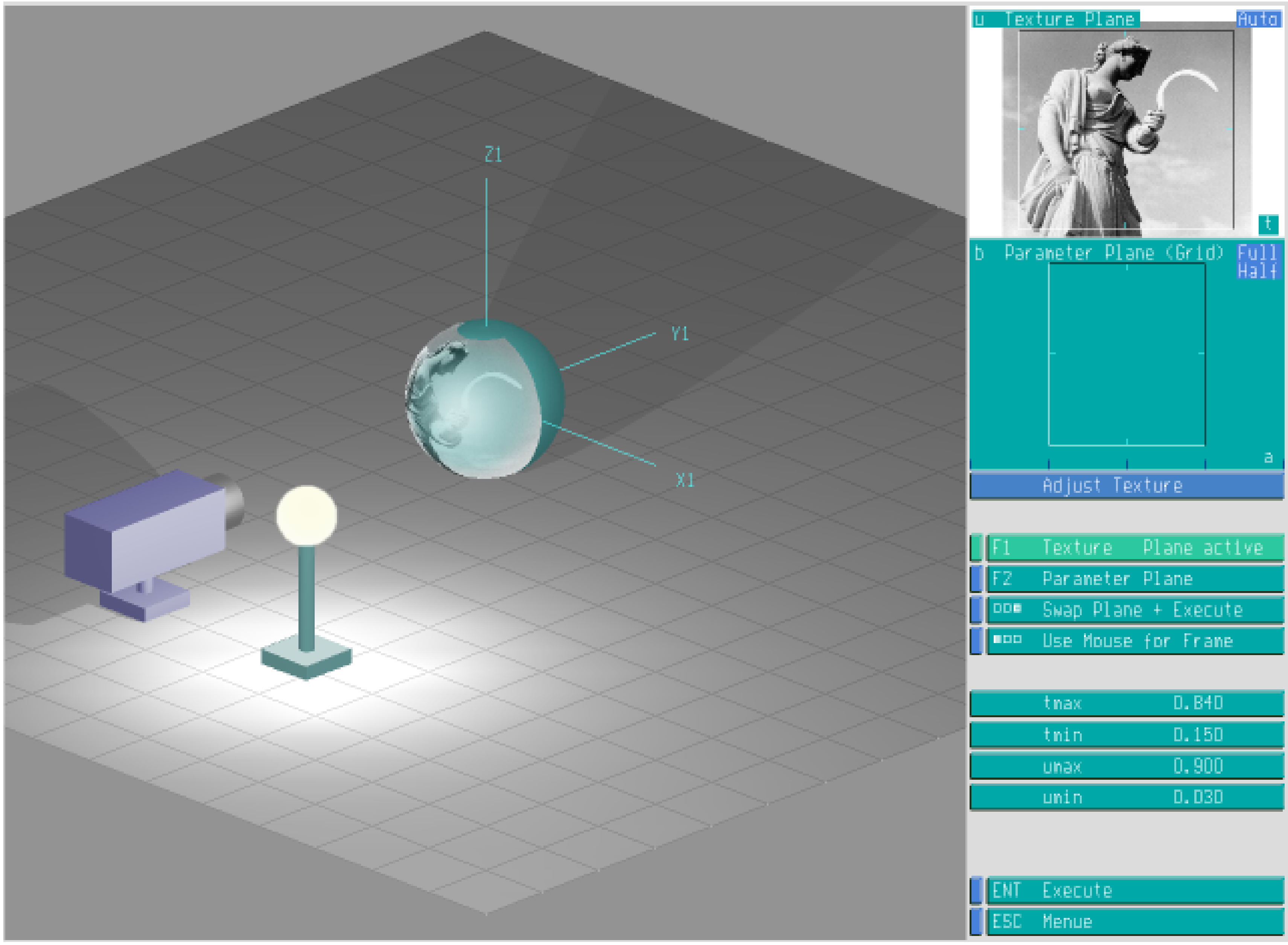






7. Texture Mapping Principles

A part of an image is mapped as a texture onto a part of the body. The relevant part of the texture image is described by a t,u-frame. The mapping area on the body is defined by a parameter plane a,b-frame.
The Icosahedron doesn´t have a parameter plane a,b so far. This has to be generated additionally. The parameter plane is either a set of sphere coordinates or a set of cylinder coordinates, which is calculated for each vertex. An alternative would be a set of Mercator cylinder coordinates.

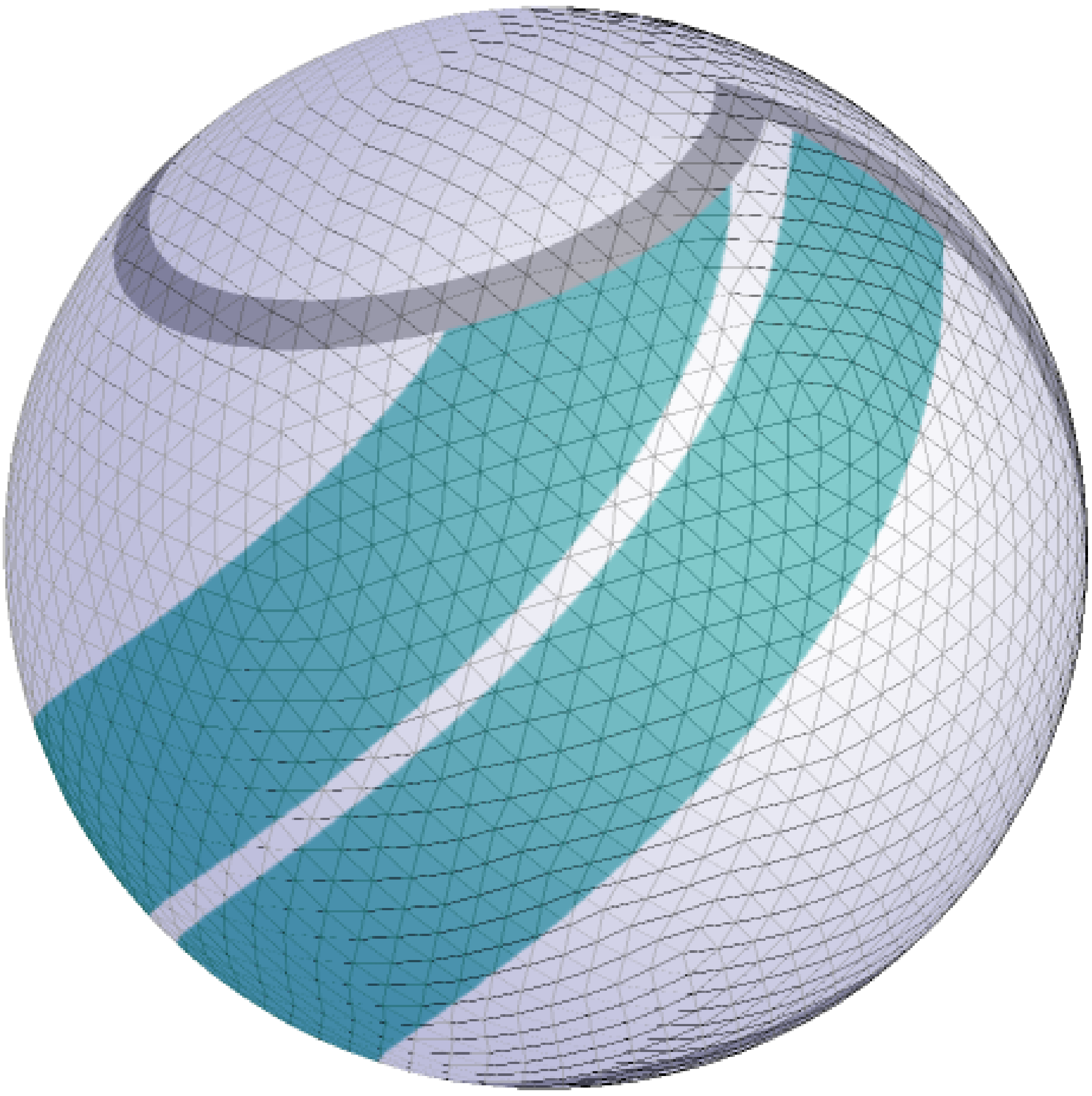


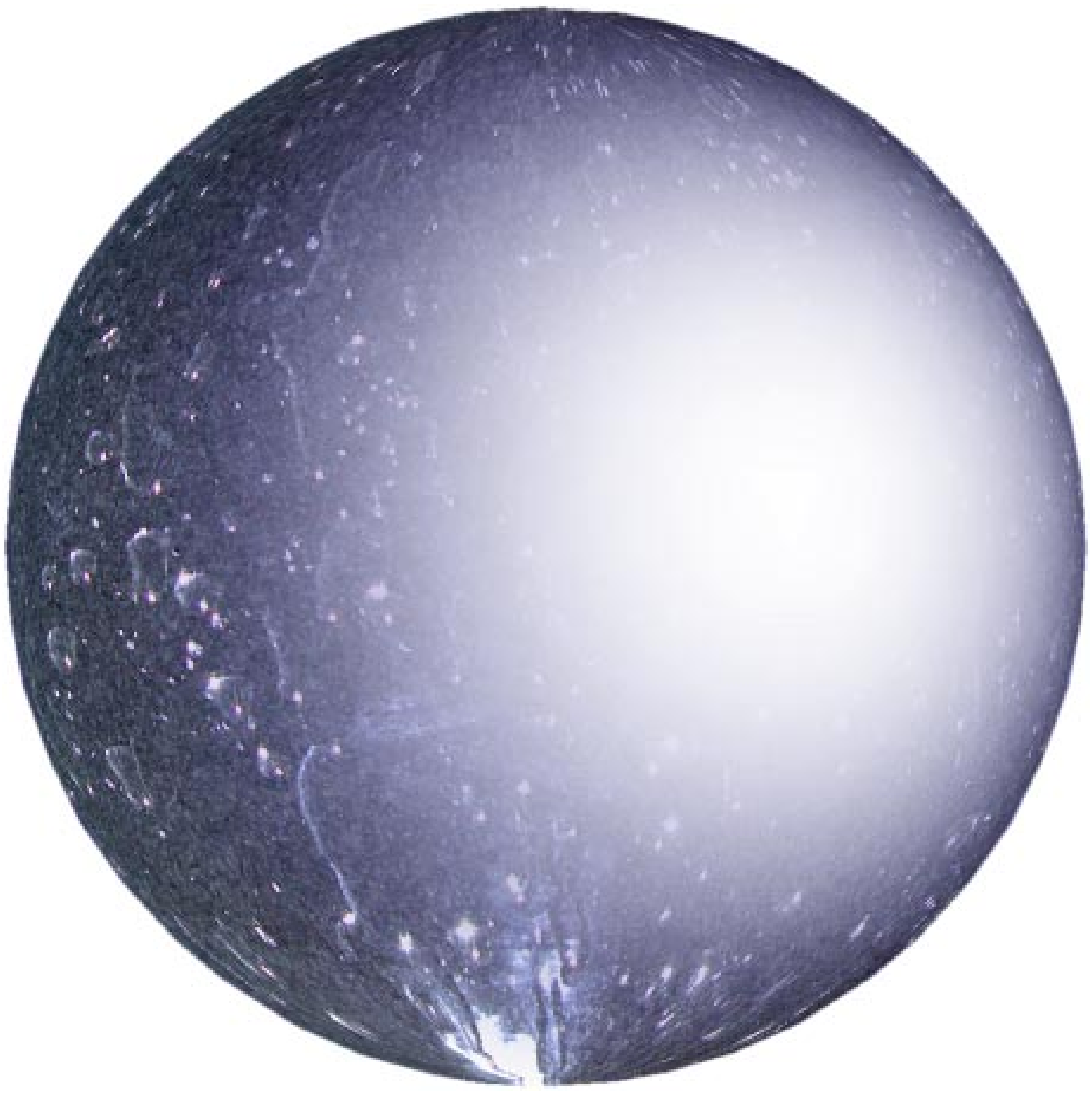

```
Procedure IcoPara;
{ Assign Parameters to vertices    }
{   a =     -pi..    +pi
    b = -0.5*pi..+0.5*pi or -1..+1 }
Var i,flag   : Integer;
    a1,a2,a3,b1,b2,b3,r1,r2,r3 : Single;
Begin
For i:=1 to tri Do
Begin
With Lt1[i]^Do
 Begin
 atangens(t1.y,t1.x,a1,flag);
 atangens(t2.y,t2.x,a2,flag);
 atangens(t3.y,t3.x,a3,flag);

 Use here one of the sets (right side)

 End;
 With Par[i]^Do
 Begin
 pa1:=a1; pa2:=a2; pa3:=a3;
 pb1:=b1; pb2:=b2; pb3:=b3;
 End;
End;
End;
```

```
Note:

atangens is the same as atan2 in C


A. Set of sphere coordinates
 r1:=Sqrt(Sqr(t1.x)+Sqr(t1.y));
 r2:=Sqrt(Sqr(t2.x)+Sqr(t2.y));
 r3:=Sqrt(Sqr(t3.x)+Sqr(t3.y));
 atangens(t1.z,r1,b1,flag);
 atangens(t2.z,r2,b2,flag);
 atangens(t3.z,r3,b3,flag);

B. Set of cylinder coordinates
 b1:=t1.z;
 b2:=t2.z;
 b3:=t3.z;
```

8. Texture Mapping Examples

The upper image shows a level 4 subdivision - 5120 triangles. The sphere is rotated by yaw and pitch. Rotation is provided because the texture is mapped to the body in body fixed coordinates. The lower sphere is not rotated, the unavoidable singularity at the poles is not visible.





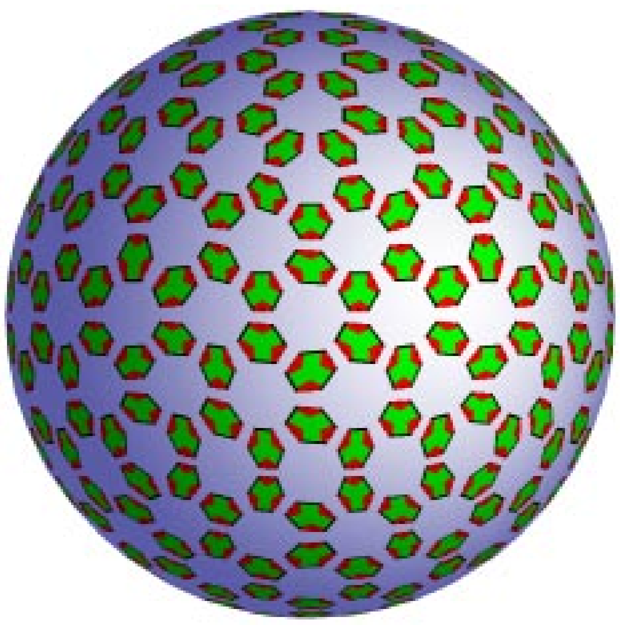
Is it worth to use subdivided Icosahedrons instead of ordinary sphere coordinates?

A sphere may be divided in 10° steps in sphere coordinates. For the latitude from -90° to +90° and the longitude from 0° to 360° we get a mesh with 648 trapezoids or 1296 triangles.

The level 3 Icosahedron subdivision results in 1280 triangles. The originally five segments are subdivided three times by two, which results in 40 angle steps of 9° .

Though we have approximately the same number of triangles, the resolution is not considerably better.
This is a surprising result, because the shrinking size of the trapezoids near to the poles in sphere coordinates lead to the assumption, that much calculation time is wasted.

Opposed to Icosahedron tessellation, the sphere coordinate tessellation can be done without storing huge coordinate tables. Furtheron, the texture mapping is much simpler in sphere coordinates.
But here we have one exception: if the texture is mapped per triangle, a structure like a golf ball, or equally distributed noise, then the Icosahedron tessellation is probably better.
In this example we have mapped one texture image to four triangles, which is slightly more complex than mapping each image to one triangle. The image itself has a special rotational symmetry.



10. References

Very effective is Google search, keyword 'Icosahedron'

The author did not use special sources

Computer Graphics by ZEFIR, Image Processing by ZEBRA

[1] Dave Eberly http://www.magic-software.com/documentation/
Plenty excellent documents for Geometry and Computer Graphics

[2] Hugo Pfoertner http://www.enginemonitoring.org/illum/illum.html
A scientific research about optimal sphere subdivisions, optimized for balanced luminance in ray-tracer applications
Many related links, a highly interesting publication

Gernot Hoffmann
May 05 / 2002 + October 23 / 2004 + November 5 / 2005
Website
Load browser / Click here