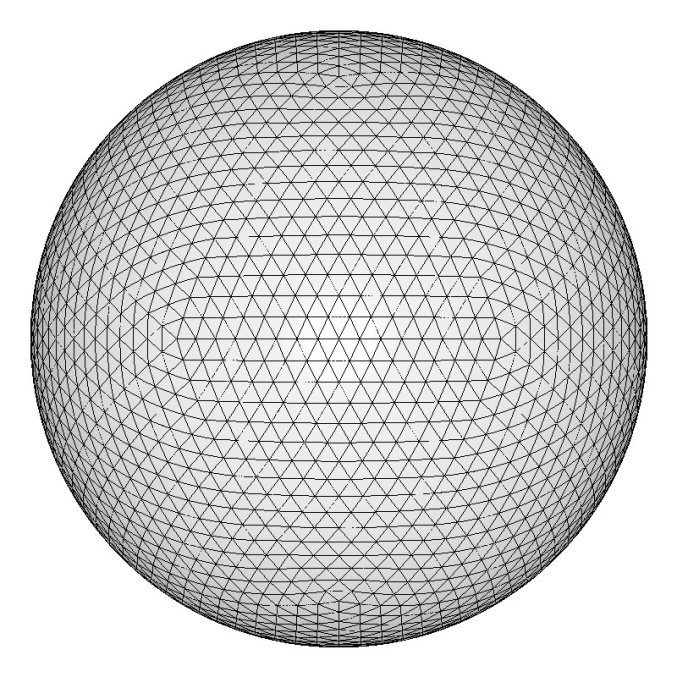
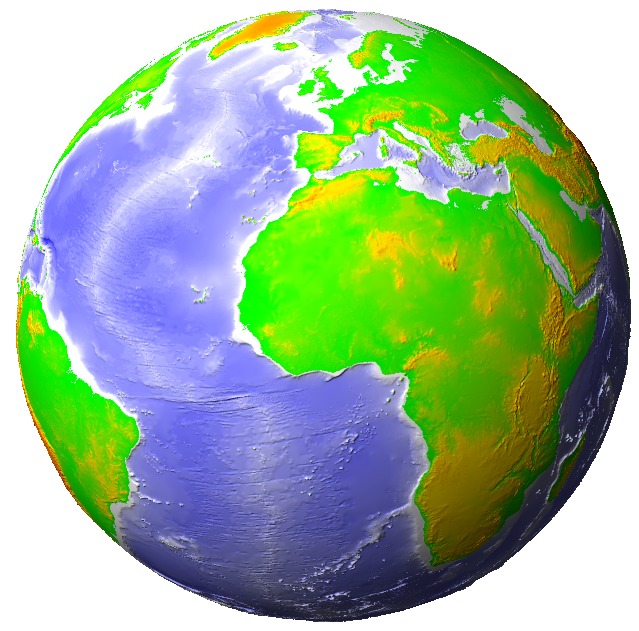
**A new geodesic grid with equally sized triangles for fast spherical finite element computing**

**** 

Maarten ’t Hart

Groningen  
The Netherlands  
[mth128@hotmail.com](mailto:mth128@hotmail.com)

18 April 2015

**A new geodesic grid with equally sized triangles for fast spherical finite element computing**

*Maarten ‘t Hart, 6 April 2015*

Abstract

This is a description and calculation method of a new type of a geodesic grid. This grid has a basis of an icosahedron, and slices it with linear cuts into smaller triangles that are equally sized, with as small variations in shape as possible. The system differs from traditional systems in the way the sphere is meshed into triangles. Instead of bisecting the ribs of the icosahedron, the 20 triangles of that icosahedron are sliced by linear cuts from a single carefully chosen projection point. It reduces calculation time and can be used as a triangular tiling system, as there are no 3-Dimensional differences between the parent triangle and their 4 higher detail child triangles combined.

In addition, a method is described to pair triangles into distorted squares. This makes the coordinate system readable, user friendly and efficient in memory storage and computing. Also, a simple method is described to find a triangle that belongs to a specific coordinate.

Keywords

Geodesic Grid, Sphere, Mesh, Triangles, Tiling, Finite Element Method.

# Introduction

The earth is a near perfect sphere. In order to compute, simulate and visualise geophysical processes, the earth is usually represented as a flat map. Various projections have different benefits. However, all these projections cause the earth to be presented in an unrealistic way. Not only does that distort the earth’s presentation, but it also requires complex calculations to compensate for the flattening. Especially sinuses and cosines increase simulation time for geophysical calculation. This usually makes spherical computing on a flat map complex and slow.

In order to perform fast computing, one may prefer to keep the earth spherical in computing. It also has the advantage of representing the earth spherical as it is. However, it is not easy to make a sphere useful in finite element computing. It would help if there would be a way to divide that sphere into smaller linear pieces that are as identical as possible, and to put that finite grid into a Cartesian system. That can be done with this new system.

For calculations, the sphere has a unitless radius of 1. The sphere has its origin in coordinate 0,0,0 in a Cartesian system. So every position in the Cartesian system belongs to a specific triangle. Every position can be converted into a vector by dividing the position by the distance of that position to the origin. Since a vector has a length of 1, every position has its vector somewhere on the surface of the sphere. This makes computing very straightforward, linear and fast.

A sphere can be meshed into equally shaped polygons. This creates a mathematical body, that is called a polyhedron. The polyhedron that comes closest to a sphere is an icosahedron, which consists of 20 identical triangles. The mathematical formulas to calculate the vertices of it in a Cartesian system are well known. It is possible to mesh it further into smaller triangles, but it is no longer possible to keep the triangles equally shaped. One must accept that these triangles differ in a way.

## Traditional geodesic grid

This is not the first geodesic grid that has ever been developed. The traditional way of meshing the icosahedron into a sphere, is by dividing each rib into halves (bisecting), and to connect the new points in order to make smaller triangles.[[1]](#endnote-1) [[2]](#endnote-2) The new points are projected onto the sphere, by dividing the coordinate of the bisect by the distance of the coordinate to the origin.

This can be repeated in order to make the mesh as detailed as one wants it to be. This way the size of the ribs can be equally sized on the edge of the original icosahedron. But all triangles in between will have a slightly different size, rib length and shape. The size variation differs per detail-level, thus it is not recommended to increase precision and detail once a mesh-density has been chosen. It is a useful grid, often used in geophysics and meteorology, but there is certainly room for improvement.

## **Optimal geodesic grid**

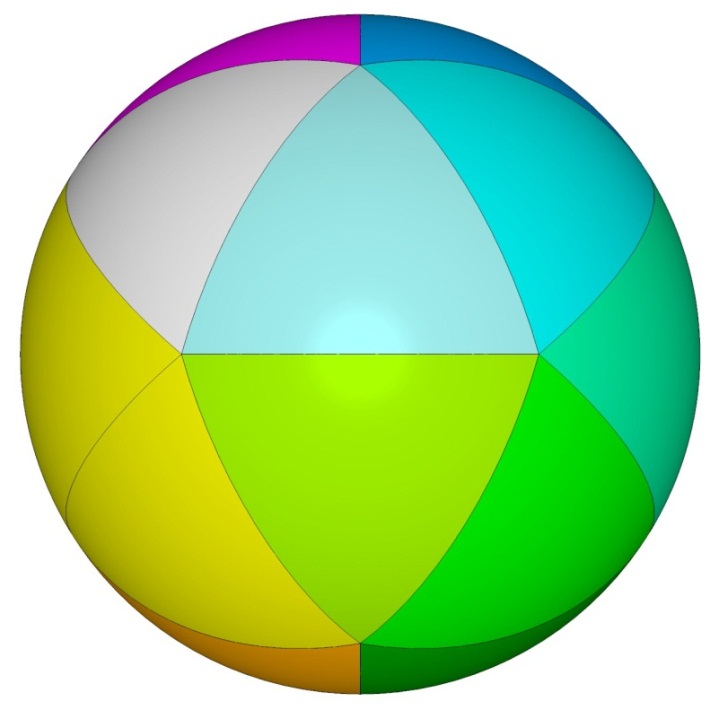
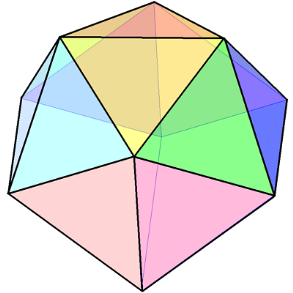
Even though we must accept that there is no way to build a sphere out of identically shaped triangles, there is a way to mesh the sphere into equally sized triangles. That can be achieved by dividing the ribs in a more sophisticated way than the traditional geodesic system. In this new system, each triangle of the icosahedron is meshed into smaller triangles, by slicing the original triangles of the icosahedron (segments) into smaller triangles by making linear cuts from a specific point. This point is called the projection point. For every triangle there are 3 identical projection points (one for each corner).

# System structure

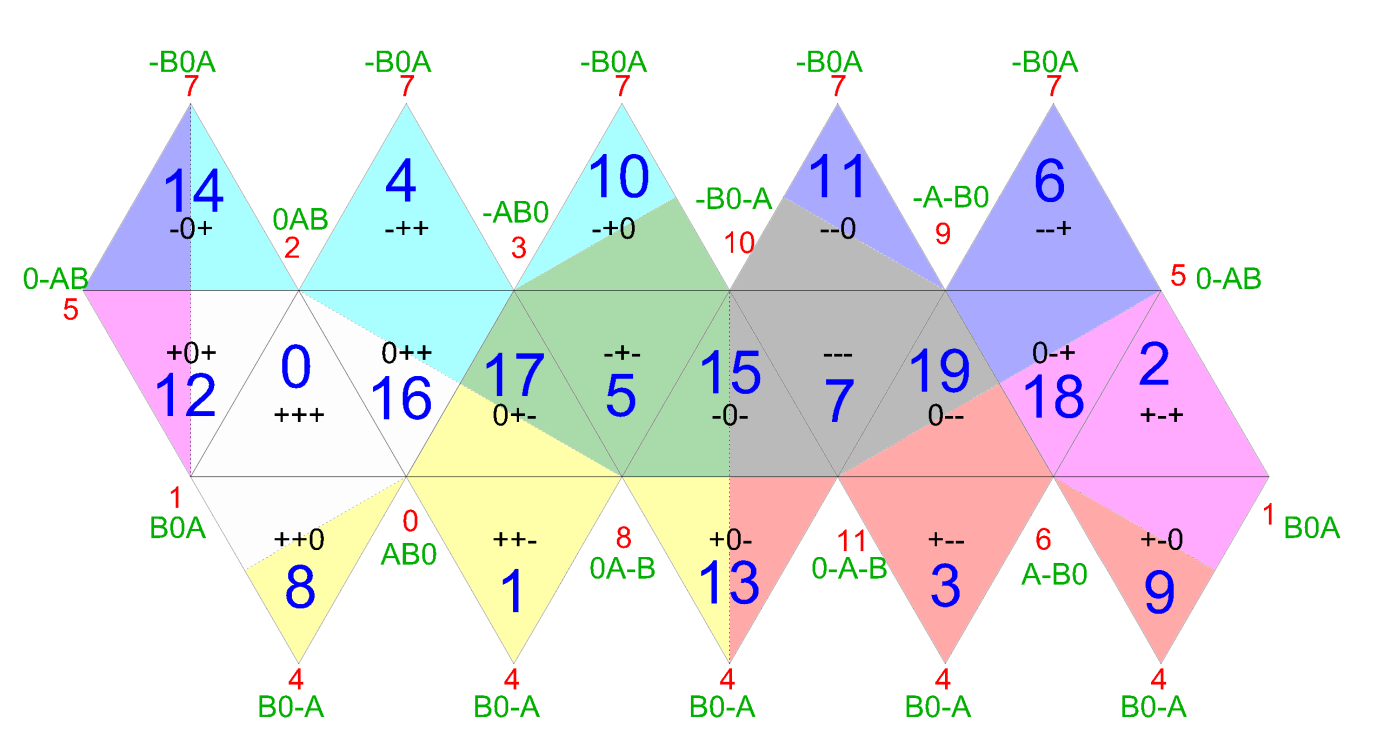
This chapter describes how the structure of the system works.

## The basis: the icosahedron

The system uses an icosahedron as a basis, see Figure 1. The centre of the icosahedron is placed in a Cartesian system at coordinate 0,0,0. Each triangle of the icosahedron has been given a logical number, corresponding to either the positive or negative values of the X, the Y and the Z coordinates (see Figure 2).



*Figure 1: Left: an icosahedron. Right: a sphere, meshed into 20 sections, using the 20 identical triangles of an icosahedron.*



*Figure 2: An unfolded icosahedron.*

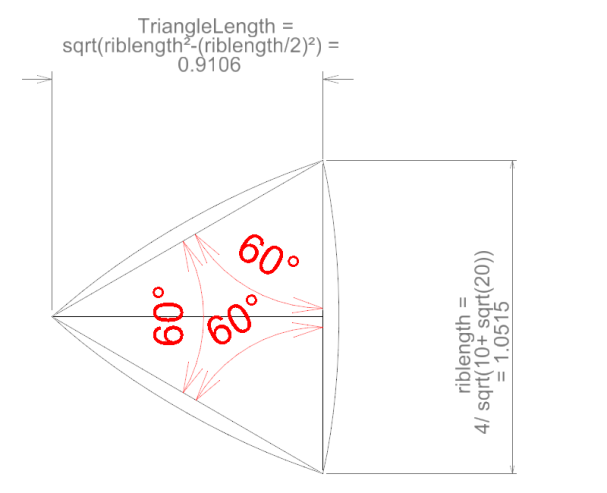
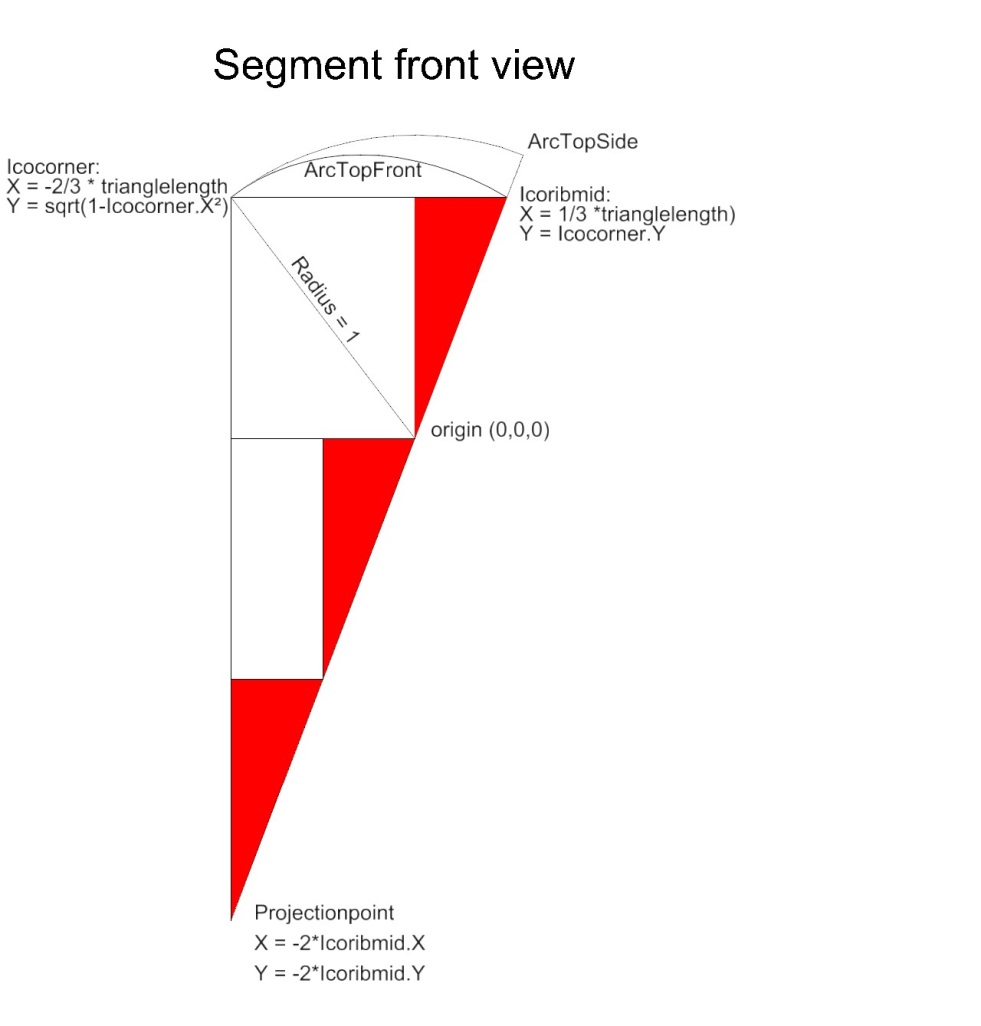
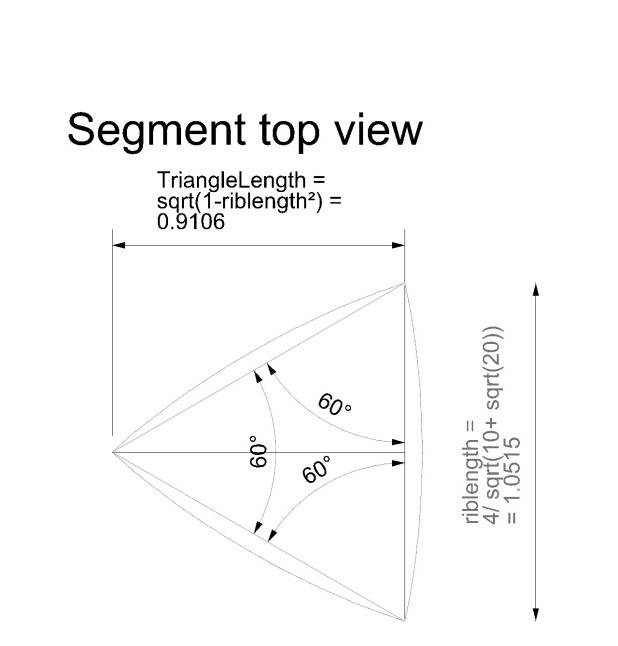
Each coordinate belongs to a specific triangle. For example: if the x, y and z values of a coordinate are all positive, the coordinate belongs to triangle 0, 8, 16 or 12 (That is within the light grey area, marked with +++). A coordinate that has –x +y and –z belongs in the green area. The 20 triangles are called “segments”. The values “A” and “B” can be calculated, and the green letters represent the coordinates for the icosahedron. This are the calculations for A and B:

Above are the formula’s to calculate the icosahedron[[3]](#endnote-3), which is the basis for this system. This icosahedron will be meshed into smaller triangles. Each triangle will have 4 “child-triangles” for each generation. So generation 0 has 20 triangles, generation 1 has 80 triangles, generation 2 has 320 triangles and so on. This is the same as traditional geodesic systems.

The size difference between the largest and the smallest triangle is measured using Bentley Microstation. The measurement shows a size variation of 0.1571 vs. 0.157 for the first generation cut. This may be due to rounding errors. If it isn’t, it is less than 0.1%, which is still a significant improvement to the traditional system[[4]](#endnote-4). For more detailed generations, Microstation does not show size variations.

## The heart of the system: the projection point

The innovation of this new geodesic system is in the concept of a “projection point”. The system cuts the sphere into triangles from that specific projection point with linear slices. In Figures 3 and 4 below is illustrated and explained how this projection point can be calculated. In order to find the projection point, one needs to look at the front view of one of the 20 icosahedron sections of the sphere, as shown in Figure 3.



*Figure 3 (left) and 4 (right): This illustration shows how the icosahedron can be calculated. The most important for this system is the projection point. From this point, the sphere is meshed into equally sized triangles.*

The three points “icocorner”, “icoribmid”, and “projection point” are the basis for the projection point plane. The icocorner point is one of the corners of the segment. The icoribmid point is the midpoint between the other two corners of the segment.

The projection point is the intersection point of the following 2 lines (as shown in Figure 3):

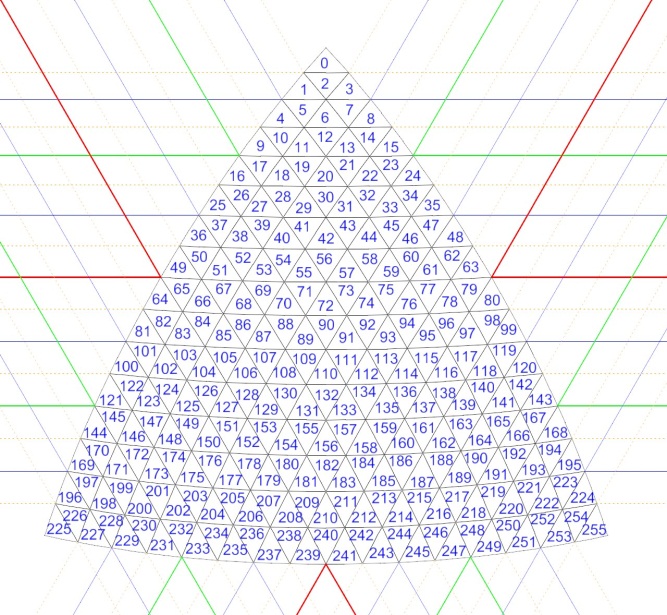
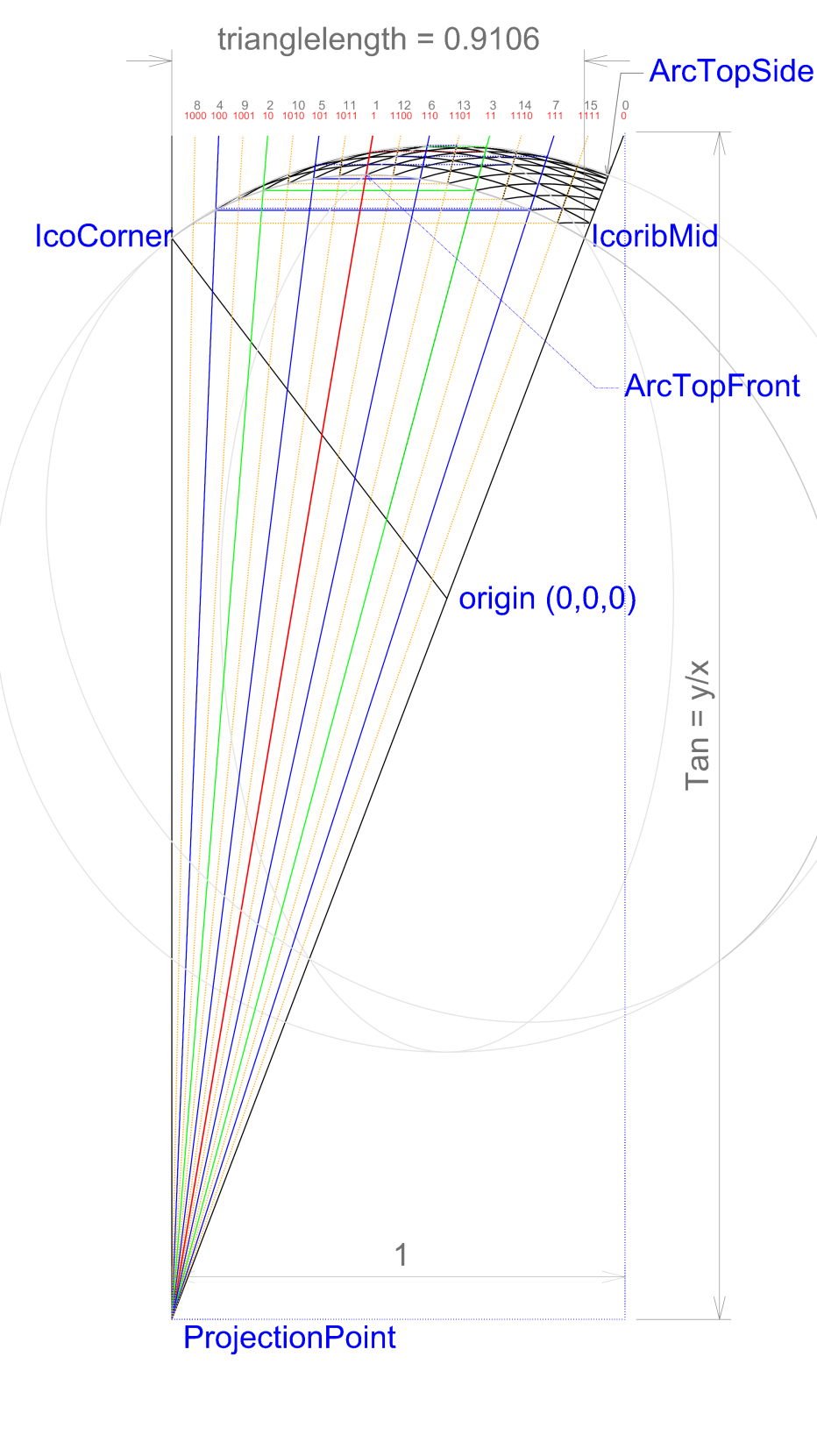
* The line that connects the origin with the icoribmid point.
* The line that is perpendicular to the line that connects the icoribmid point with the icocorner point, intersecting at the icocorner point.

## Tiling

The new system also allows tiling. Increasing detail level adds new cuts to the previous detail level, dividing a parent tile into 4 equally sized child tiles. The combined shape of these 4 child tiles is identical to their parent tile. Therefore, detail level can be increased “on the fly”.

## The meshing phase

Here is an illustration and description of how to mesh the sphere into triangles.

**

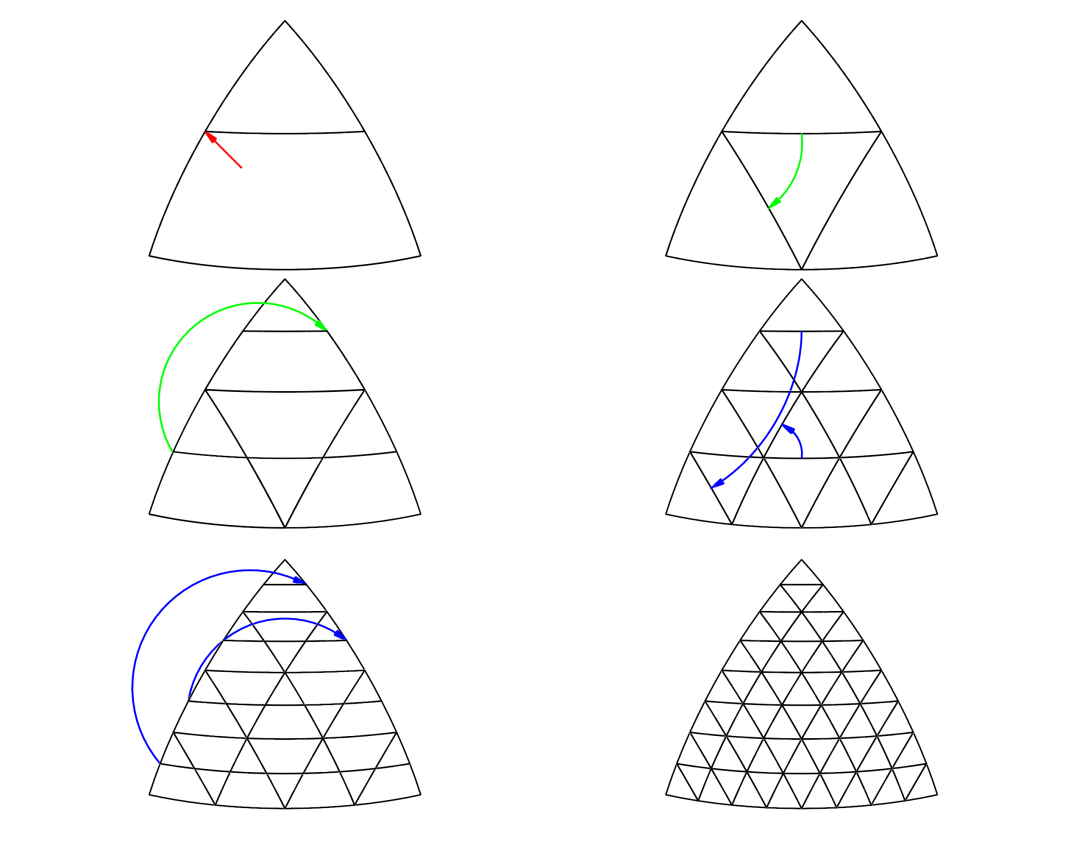
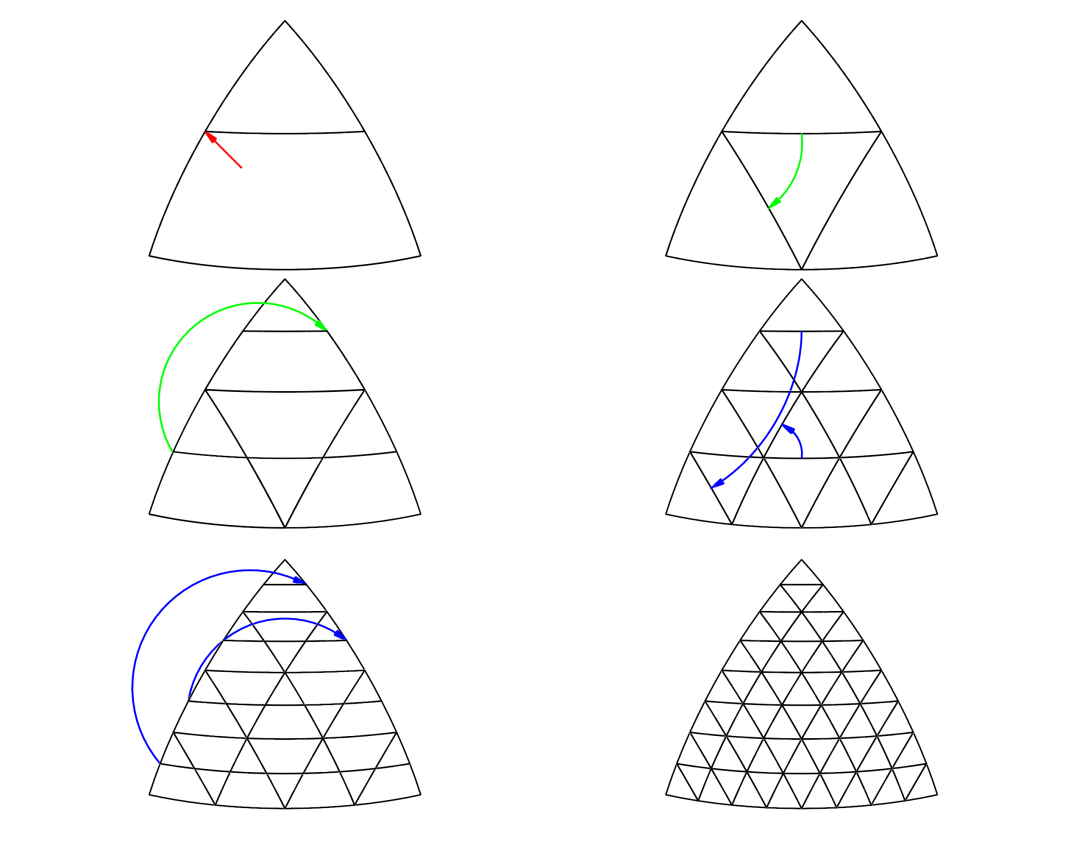
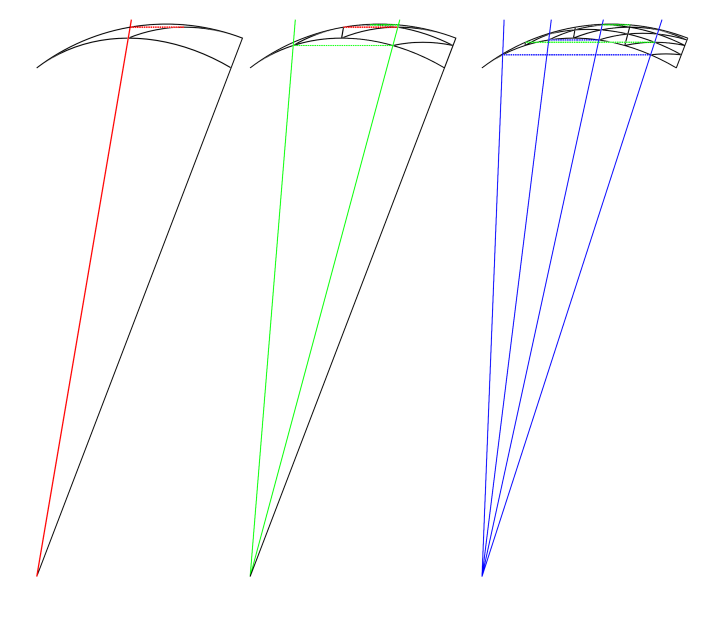
*Figure 5 (left, side view) and 6 (right, top view): meshing process of the segment. The segment is meshed into smaller triangles from the projection point.*

Note to Figure 5: it looks like the side arc of the segment is cut through the exact midpoint of the split arc each time (just like traditional geodesic grids with the bisecting type meshing process), but that is not the case (except for the first generation cut -red line-). Striking through the midpoints of the side arc causes the triangles not to be cut correctly. It will leave tiny areas either not included in any triangle or overlapped by multiple triangles, and also causes the sizes to be not equal for each triangle.

The start value for the calculation is the triangle length from Figure 4. With this value, it is possible to calculate the coordinates marked in Figure 5. This includes the starting coordinate for the meshing process, the “ArcTopFront” point.

To get a vector of a point, this is the formula:

The ArcTopFront point is the first strike through point of the red line.



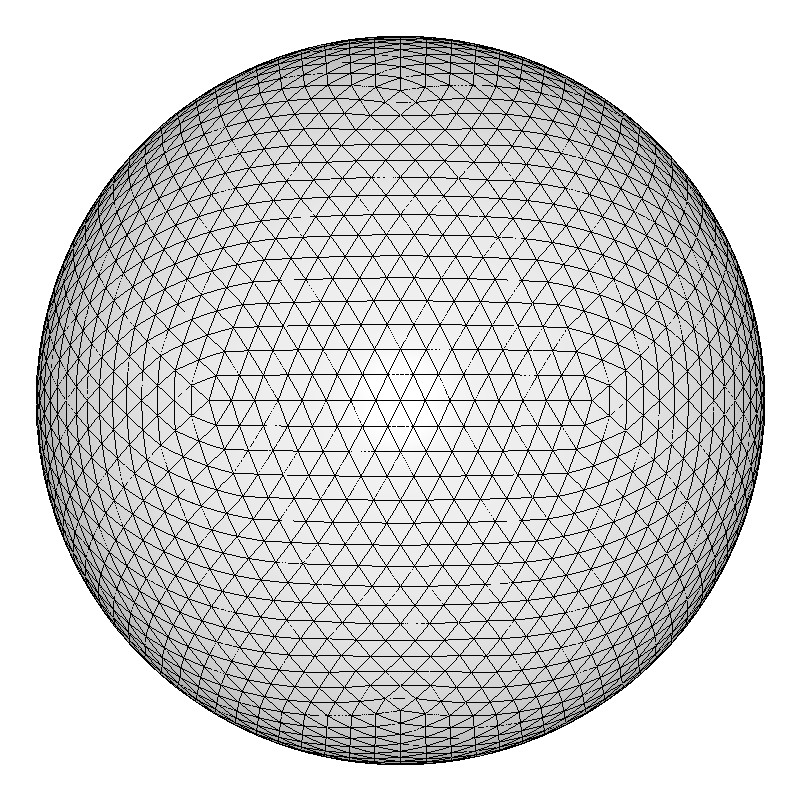
*Figure 7 Building scheme for the geodesic grid*

In order to make this a perfect geodesic grid, the following steps must be performed. The cut lines through the segment are shown in Figure 5 for the first 4 generations. Before meshing, we set up an array of tangents of these cut lines, using the method shown in Figure 7.

1. We start with the side view. The first cut through the segment is through the “ArcTopFront” point. It is the first generation cut, the red line.
2. The intersection point of this cut line with the edge of the sphere must be calculated: this is the point where the line “exits” the sphere in the side view.
3. Next step is in the top view. The new point (2) must be rotated 120 degrees. That point is now the primary strike-through point for the next cut.
4. Back to the side view: the right green line strikes through that primary point.
5. The green cut has an intersection point with the front-arc. This point must be calculated.
6. Back to the top view. The new intersection point (5) must be rotated 120 degrees to get a secondary strike through point.
7. Back to the side view: the left green cut strikes through that secondary point.
8. For both green lines, steps 2 to 7 must be repeated in order to get the 4 blue lines.
9. For all four blue lines, steps 2 to 7 must be repeated in order to get the 8 orange lines.
10. Keep on repeating this process until the desired detail level is achieved.

All these striking lines have got a unique tangent, which can be calculated by y/x. This tangent must be stored in a sorted array. The tangents are used to calculate all slicing planes through the sphere. All grid points are the intersection point of 2 slicing planes and the sphere. The computer program can also use the tangents later for finding the right triangle to a specific coordinate. The best way to store these values, is by using a binary search method. The tangent of the red line must be stored at index 1 in the array, the tangents of the green lines sorted from small to large at index 2 to 3 in the array, the tangents of the blue lines sorted from small to large at index 4 to 7, and so on. This way, a binary search to the desired generation can be easily performed later.

Note: the tangent array is independent of the grid itself. It is possible to generate a very deep generation array of tangents, while the grid is meshed into a less detailed generation.

****

*Figure 8: The full grid in generation 4.*

# Coordinate system

This chapter describes how the coordinate system of the new geodesic grid works.

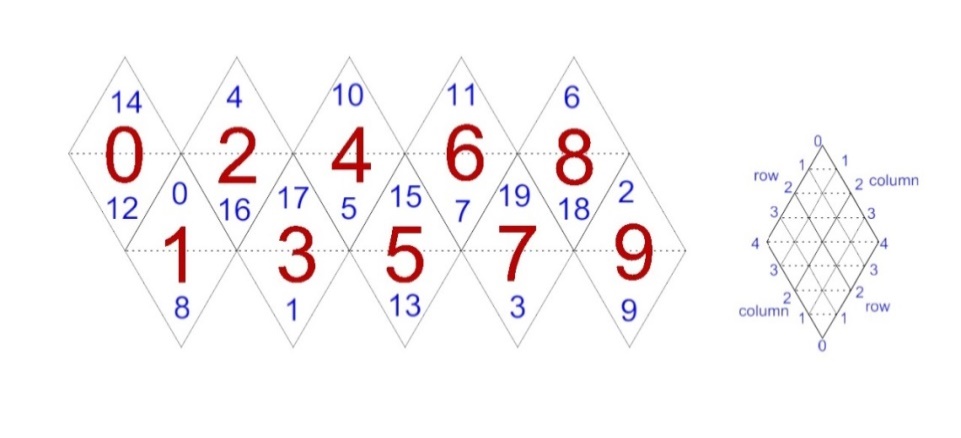
## Simple coordinates by using paired triangles/distorted squares

The disadvantage of geodesic grids, is that coordinates are hard to read. Coordinates in the system consist of 3D vectors. The commonly used alternative coordinate systems (geographical and projected) have coordinate systems expressed in 2D values. Latitude/longitude (or N/E) for geographical, and easting/northing (or X/Y) for projected systems. The most direct way of giving a coordinate is by numbering the triangles of the gird, but it is quite unclear for the user to know where it is by seeing a number of a triangle (see Figure 6). Triangles are not easily expressed in rows/columns. In most cases, geodesic systems are expressed in hexagons. This is a better way of producing readable coordinates.[[5]](#endnote-5) However, this system has the need for dead cells, which takes more computer memory and a bit unclear amount of hexagons for each detail level.

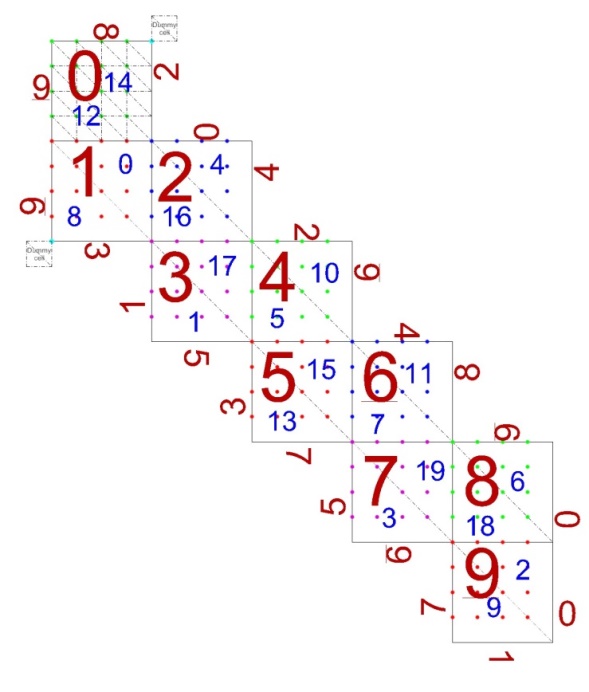
Another way to express the grid is to use “paired triangles”. A triangle has three vertices that set the boundary. When one combines 2 triangles, there are 4 vertices that set the boundary. This can be considered a square, so that it allows for 2D representation and row/column display. They are not really square shaped, so it is more correct to call them distorted squares. One of these vertices can be selected to be considered the base point of the paired triangle. The other three points are then the base points of the adjacent paired triangles.

This can already be done with the very first detail level of the system: the original icosahedron. It has 12 vertices and 20 triangles. When combining the triangles to paired triangles, one can display it as if the icosahedrons consists of 10 distorted squares. With one single vertex (i.e. point) considered the base of each paired triangles, 10 points are directly related to distorted squares. The 2 remaining of the 12 vertices are not the primary point of any square. These are 2 separate “dummy points” that are used to close the system. Within the system, these two points are placed at the end of the array.

With each increase of detail level, the amount of paired triangles is multiplied by 4. So the first generation (detail level) consists of 40 paired triangles, and 42 points. The two points more than triangles, are the same dummy points as in the previous generation. Generation 2 has 160 paired triangles, and 162 points, again with the same 2 dummy points.



*Figure 9: interpreting the geodesic triangle grid as paired triangles for row/column representation.*



*Figure 10: interpreting the geodesic triangle grid as distorted squares. This shows the 162 coordinates of a generation 2 geodesic grid.*

## Simple coordinate display

The first 10 paired triangles should be considered 10 separate grids. These 10 grids must be programmatically sealed together. Within these 10 grids, one can consider the tiles (or cells) as rows and columns. And very important, the generation (i.e. detail level) should also always be mentioned.

Generation 8 has 256 ( = 28) rows and columns per base paired triangle. This has 65536 “distorted squares” per base paired triangle. So the mesh has 655360 distorted squares, which is 1310720 triangles.

A proper coordinate can be: Generation: 8, Paired triangle: 3, Row: 170, Column: 55. Or simply 8, 3, 170, 55. This makes coordinates very readable for the user. Especially because the grid number is always a value between 0 and 9, and the generation value never reaches up high. In case of generation 34, the earth would be meshed in triangles that have an area of less than 1mm2.

To calculate which point this is in the array, one must calculate:

* 28 = 256 rows or columns per paired triangle
* 2562 = 65536 distorted squares per grid
* Index = 3 \* 65536 + 170 \* 256 + 55 = 240183.

The four neighbouring distorted squares for coordinate 8, 3, 170, 55 are:

* 8, 3, 169, 55
* 8, 3, 171, 55
* 8, 3, 170, 54
* 8, 3, 170, 56

This does not only make it easy for the user, it is also very simple for the computer, as it is just an increment or decrement of 1, and a couple of multiplications and additions. The only case when it gets a bit tougher for the user, is in case of passing the boundary of the grid to the next grid.

The possibility of increasing and decreasing detail level, shows one other advantage over the hexagon method: finding child or parent triangles require less calculation time, because it only requires one shift operator in the row and column value. But the hexagon method is also compatible with the new meshing system, so in case one would only want to update the meshing system, it is not required to use the new coordinate system as well.

## High detail mesh of a specific area

Since this grid is built by planes that slice the sphere into triangles, a useful feature is that the row and column number is equal to the slicing plane index. This makes it possible and very easy to mesh a very specific area of the sphere into very high detail, without it requiring the entire sphere to be meshed into the same high detail level.

### Effect of increasing detail level on the coordinate system

In case of increasing detail level with 1 generation, each distorted square is cut into 4 distorted squares. To find the 4 coordinates of the child distorted square, the generation must be incremented with 1, and the row and column value must be multiplied by 2. That gives the coordinate of the top left distorted squares. For the other three, add 1 to the row and/or column value. Example, for coordinate 8, 3, 170, 55 the child distorted squares are:

* 9, 3, 340, 110
* 9, 3, 341, 110
* 9, 3, 340, 111
* 9, 3, 341, 111

To find a parent coordinate, just do the opposite: generation -1, and divide the row and column by 2 (odd values rounded down). So the parent coordinate of coordinate 8, 3, 170, 55 is 7, 3, 85, 27.

### Decreasing display detail level, without decreasing computing detail level

In case one would like the computer to be fast in displaying a very detailed geodesic grid, one may like to display the grid in lower detail level than the grid itself. In order to do that, one must iterate through the array with a larger step size of the row and column values. This must be done with step size: 2(base generation – display generation). So if the base generation is 8, and the display generation is 5, one must iterate through the rows and columns with step size 8. If the base generation is 8, and the display generation is 7, the row/column iterator step size is 2.

## Decimal coordinates

In order to store precise coordinates, one can make a choice what to store. The quickest storage may be storing the three T values of the coordinate. However, in case of X = K, T will be infinite because of a division by 0.

The most readable and probably best choice, is to store two Xt values (see Figure A.1) of the first two (top)arcs of the segment. This value is always between K and -1/2 K. One can add the value of the segment to the first value of the coordinate. This is the most compact storage of a precise geodesic coordinate. To regain the Cartesian coordinate, calculate the slicing planes from the two values, intersect the planes to get a straight line, and find the point on that line that has a distance of 1 to the origin (i.e. intersection point of the two slicing planes with the geodesic sphere). Or just recalculate the T values from the Xt values, which is much easier.

Example coordinate: we have a point that has an Xt value of 0.212 on the first TopArc, and an Xt value of -0.52 on the second arc, on segment 15. In that case the coordinate A, B is 15.212, -.52. For the segment numbers see Figure 2.

## Procedure to find the triangle that belongs to a cartesian coordinate

In order to find the right triangle that belongs to a specific coordinate, the following steps are performed by the computer program. Also see Figure 11 to 14:

1. The coordinate is divided by its distance to the origin, in order to get its vector.
2. The x, y and z of the coordinates are checked for being either positive (+) or negative (-) to define to which area of the sphere the coordinate belongs. (see Figure 2)
3. The x, y and z value are further analysed to define to which of the 4 remaining sections the coordinate belongs.
4. The tangent of the vector to the three projection points of the segment are calculated, by projecting the vector into the projection point plane (Figure 12). In that view, the y-distance to the projection point must be divided by the x-distance to the projection point. That gives a tangent value for each of the three projection points.
5. Those three tangent values are analysed by a binary search method, to define the column, row and third index values for the grid. The binary search is necessary to find an index value from a tangent (T) value. The binary search is very simple. Start with Index = 1, and perform the following function:

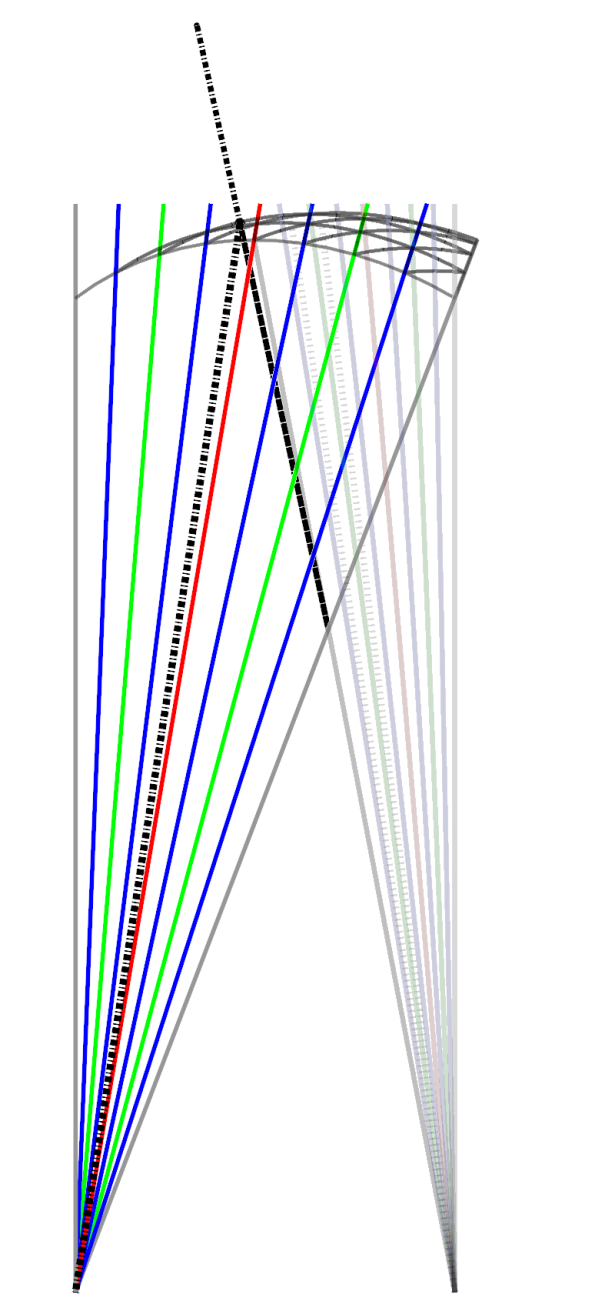
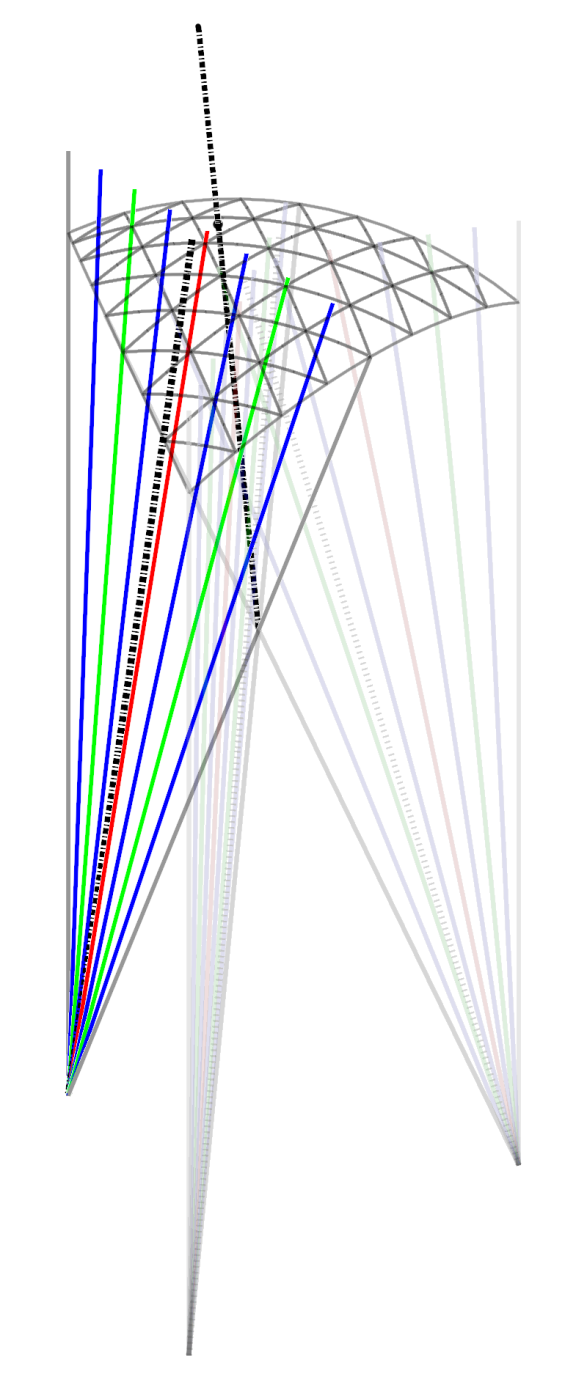
*if (T < TangentArray[Index])   
 Index = Index\*2;*

*else*

*Index = Index\*2+1;*

Do this step once for each desired generation. Eventually, subtract the maximum index value of that generation (i.e. remove the front bit) from the resulting index value, to get the correct index value. This value represents the column or row index.   
This only works if the calculated tangents of the grid are stored in this way identically of course. The sorting of the tangents has been described earlier in this paper.

1. The three index values that result from the binary search can be put in a simple calculation to find the exact triangle the coordinate belongs to. (Figure 14)



Point of interest

Projection point

Vector of Point of interest

Tangent (y/x)

Point of interest

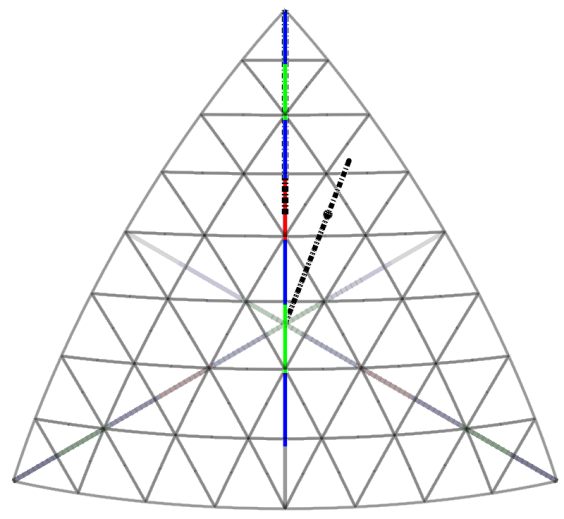
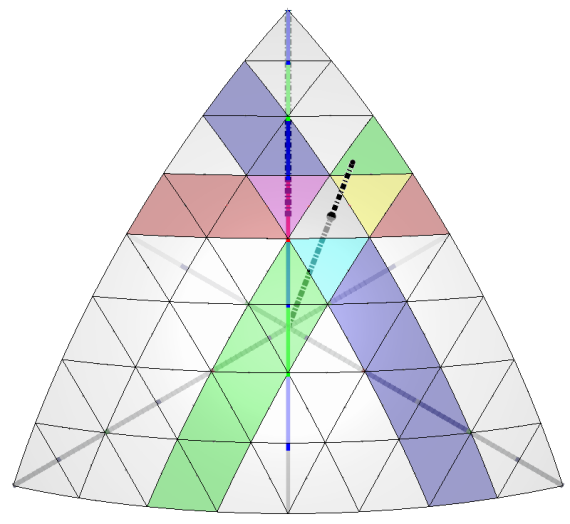
Vector of Point of interest

Origin

x

y

*Figure 11 (left): Isometric view & Figure 12 (right) side view of the triangle search method.*

Tangent column

Tangent row

0

1

2

4

5

6

7

3

7

6

5

3

4

1

2

0

3rd tangent

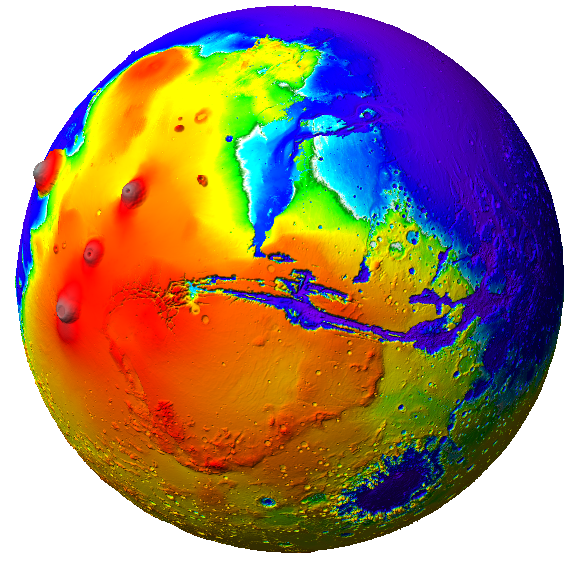
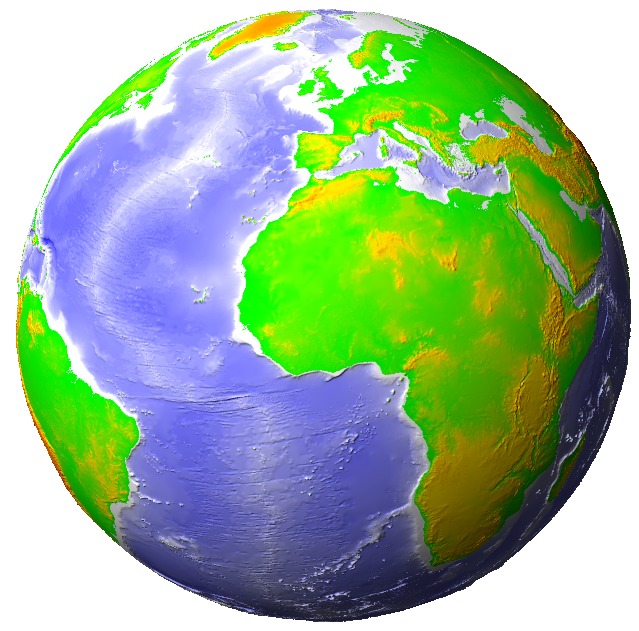
Triangle of interest

Point of interest

Vector of Point of interest

*Figure 13 & 14: top view of the triangle search method. In this example, the triangle of interest is the top triangle in distorted square 1x8+2 = 10 of the example grid.*

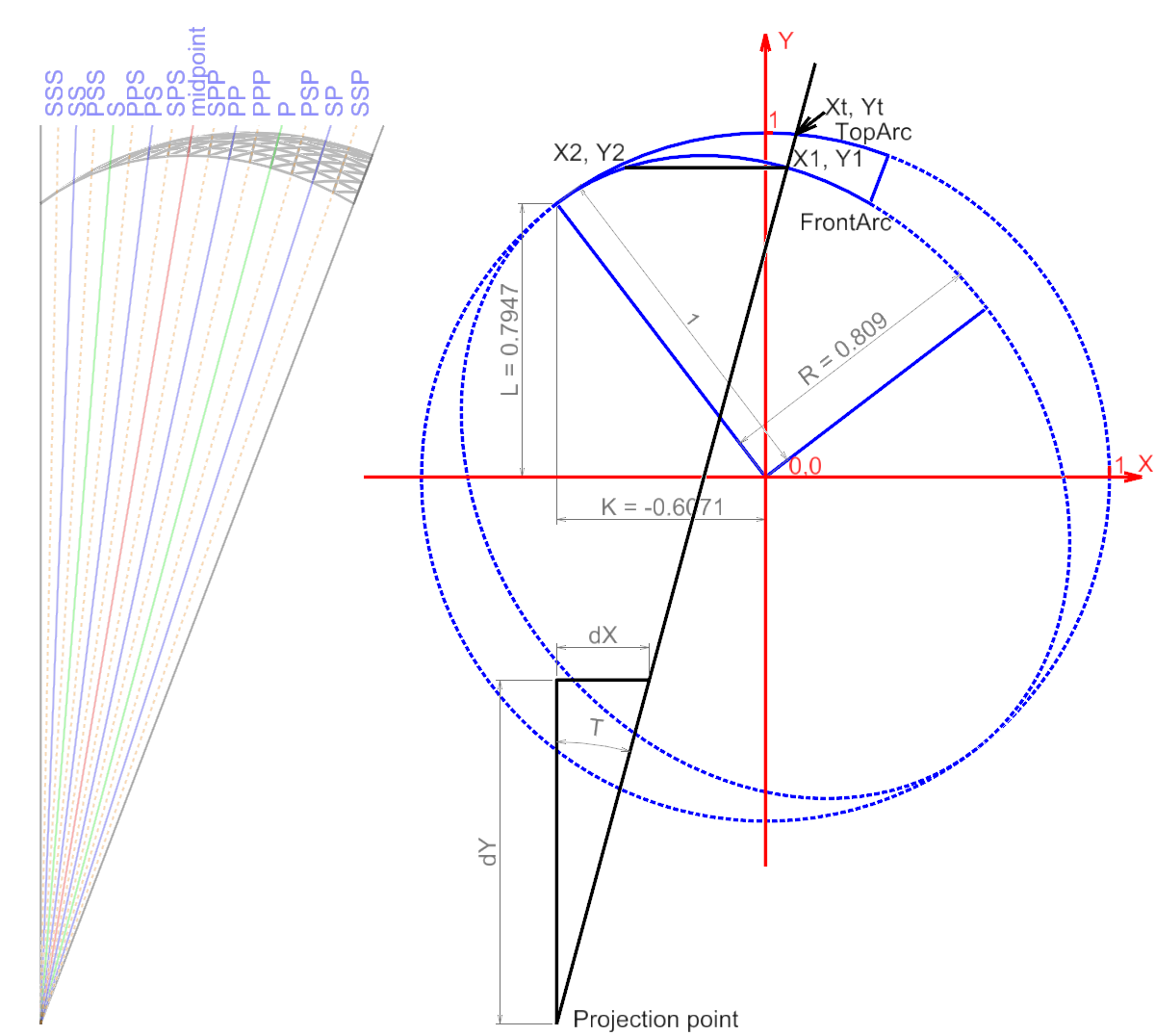
Below is an example image of the geodesic grid used in a computer program.



*Figure 15: the geodesic grid used to visualize a height map of the earth[[6]](#endnote-6) (left) and a height map of Mars[[7]](#endnote-7) (right) using 5242880 identically sized triangles. (Altitude data scaled 10 times relative to planet radius.)*

# Advantage of this system

This new geodesic system does not use any angles, sinuses or cosines. It only uses linear vector calculations and a binary search method. That makes the system very fast. The system calculates and presents the earth as a sphere made out of equally sized triangles. Also the way of adding detail allows this to be used for tiling. Detail level can be increased for a specific area on the fly. This results in straightforward, fast and detailed spherical calculations. Pairing triangles into distorted squares makes the coordinate system both readable for end users and fast for computing. In addition, a straight forward method is described to find a specific triangle for a specific coordinate.

Appendix: constant and formula overview 

*Figure A.1: Constant and variable definitions*

**Constants:**

|  |  |  |
| --- | --- | --- |
|  | |  |
|  | |  |
|  |  |  |

**Formulas:**

**X from Y on the FrontArc and inverse**

**X1 and Y1 from a known T:**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

**LatLon (N, E) to XYZ:**

|  |  |  |
| --- | --- | --- |
|  |  |  |

**XYZ to LatLon (N, E):**

**Calculation sequence of “T”**

To = previous “T”.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | |  |
|  | |  |
|  | | |
|  |  |  |

The first T as input, is that of the midpoint of the FrontArc, which is the TSecondary of Icoribmid (Figure 3). All other T’s are following from a sequence of “Primary” and “Secondary” equations of that first T. Every TPrimary and TSecondary can be used as the new value for To in order to find the following TPrimary and TSecondary values for the next generation.

Example: TPSP = Primary(Secondary(Primary(Secondary(IcoRibMid.Y/IcoRibMid.X))))

**Finding the T values for a rotated segment**

The equations in this example assume the segment to be oriented so that the segment is seen from the side, and the IcoCorner and IcoRibMid have Z=0 and the X values are identical for these points. In the geodesic system, all segments are 3 dimensionally rotated. In order to keep calculations the same for the entire system, the calculations can be done by performing the same calculations with vectors.

The X-vector is the vector of the line that connects the IcoRibMid point and the IcoCorner point. The Z-vector is the vector of the line that connects corner point 2 with corner point 3 of the segment. The Y-vector is the normal vector of X-vector and Z-vector:

The “vector distance” to any given point is easy to calculate. Example:

So instead of using a direct X or Y value, one can calculate the X-distance and Y-distance using X-vector and Y-vector The rest of the calculations are identical to the system that is oriented to the Cartesian system as shown in Figure A.1.

**References**

1. *Modeling the Atmospheric General Circulation Using a Spherical Geodesic Grid: A New Class of Dynamical Core,* Todd D. Ringler, Ross P. Heikes, and David A. Randal. Department of Atmospheric Science, Colorado State University, Fort Collins, Colorado [↑](#endnote-ref-1)
2. *Icosahedral Discretization of the Two-Sphere*, Author(s): John R. Baumgardner and Paul O. Frederickson, Reviewed work(s). SIAM Journal on Numerical Analysis, Vol. 22, No. 6 (Dec., 1985), pp. 1107-1115 [↑](#endnote-ref-2)
3. The icosahedron, <http://mathworld.wolfram.com/Icosahedron.html> [↑](#endnote-ref-3)
4. Creating a geodesic grid, <http://kilodot.com/post/3690269953/creating-a-geodesic-grid> [↑](#endnote-ref-4)
5. Grid coordinates for a hexagonal mesh, <http://kiwi.atmos.colostate.edu/BUGS/geodesic/text.html> [↑](#endnote-ref-5)
6. Gridded bathymetry data: the gebco 1 minute grid, <http://www.gebco.net/data_and_products/gridded_bathymetry_data/> [↑](#endnote-ref-6)
7. Mars Global Surveyor/Mars Orbiter Laser Altimeter (MOLA) data. Mission Experiment Gridded Data Records, <http://geo.pds.nasa.gov/missions/mgs/megdr.html> [↑](#endnote-ref-7)