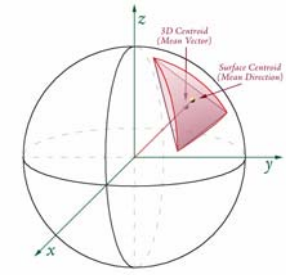
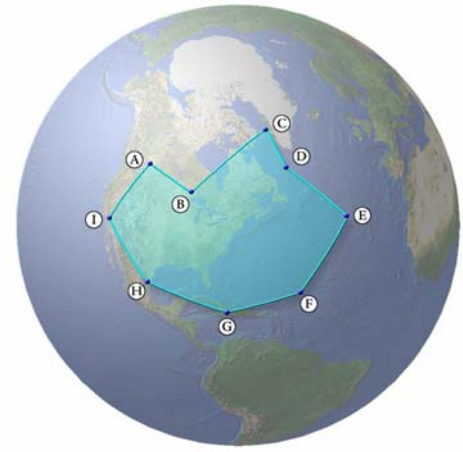
Geodesic Tools

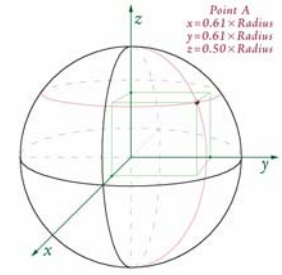
A selection and expansion of Geodesic Functions from

Tools for Graphics and Shapes

By Jenness Enterprises







Last updated 30 December 2008

Jeff Jenness

Jenness Enterprises

About the Author

Jeff Jenness is an independent GIS consultant specializing in developing analytical applications for a wide variety of topics, although he most enjoys ecological and wildlife-related projects. He spent 16 years as a wildlife biologist with the USFS Rocky Mountain Research Station in Flagstaff, Arizona, mostly working on Mexican spotted owl research. Since starting his consulting business in 2000, he has worked with universities, businesses and governmental agencies around the world, including a long-term contract with the United Nations Food and Agriculture Organization (FAO) for which he relocated to Rome, Italy for 3 months. His free ArcView tools have been downloaded from his website and the ESRI ArcScripts site over 200,000 times.

NAME: Geodesic Tools

Install File: Geodesic_Tools.EXE
Last modified: December 30, 2008

TOPICS: Sphere, Vincenty, Geodesic, Great Circle, Spherical Triangle, Polygon, Centroid, Center of Mass, Vector, Area, Haversine

AUTHOR: Jeff Jenness
Wildlife Biologist, GIS Analyst
Jenness Enterprises
3020 N. Schevene Blvd.
Flagstaff, AZ 86004 USA

Email: jeffj@jennessent.com
Web Site: http://www.jennessent.com)
Phone: 1-928-607-4638

Description: This extension provides several tools for calculating surface measures (coordinates, distances, areas) on a variety of spheres or spheroids, with specialized functions optimized to assist planetary researchers with planetocentric / planetographic conversions. All tools are available at the ArcView license level.

This extension also includes a tool to wrap existing geographic datasets around longitude ranges of -180° to 180° or 0° to 360°.

Note: These tools will eventually be wrapped in to the larger "Tools for Graphics and Shapes" extension, which will be available at http://www.jennessent.com/arcgis/shapes_graphics.htm.

Output: Several tools produce point, multipoint, polyline or polygon shapefiles. Some geometry tools produce new shapefiles while others add new fields to attribute tables.

Requires: ArcGIS 9.1 or better This tool has only been tested on Windows 2000 and XP.

Revision History None yet

Recommended Citation Format: For those who wish to cite this extension, the author recommends something similar to:

Jenness, J. 2009. Geodesic Tools: Extension for ArcGIS. Jenness Enterprises.

Table of Contents

Description and Installation

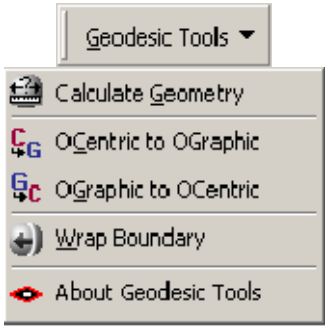
Install the Geodesic Tools extension by double-clicking on the file “Geodesic_Toos.EXE” and following the instructions. The installation routine will register the Geodesic_tools.dll with all the required ArcMap components.

The default install folder for the extension is named “Geodesic_Tools” and is located inside the folder “Program Files\Jennessent”. This folder will also include some additional files and this manual.

These tools are installed as an extension in ArcMap, but it is a type of extension that is automatically loaded. You will not see this extension in the “Extensions” dialog available in the ArcGIS “Tools” menu. It is not dependent on any other extensions or any ArcGIS license level.

After installing the extension, you should see the following new toolbar in your map (it may also be embedded in your standard ArcMap toolbars, rather than as a standalone object):

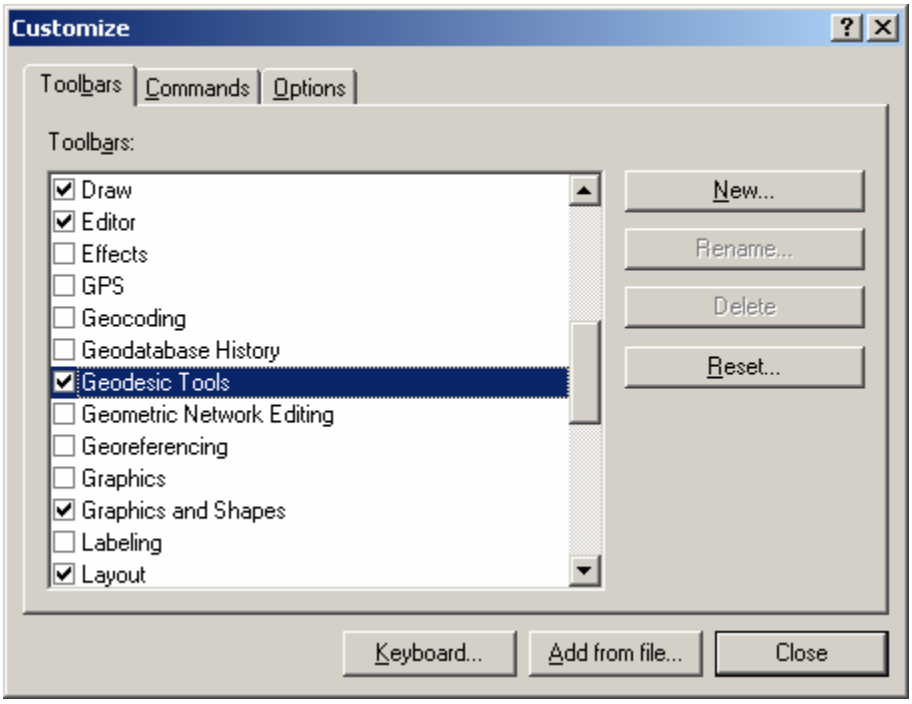
Click the “Geodesic Tools” menu located on this toolbar to see the available tools:



If you do not see this toolbar, then open your “Customize” tool by either:

1) Double-clicking on a blank part of the ArcMap toolbar, or
2) Clicking the “Tools” menu, then “Customize”.

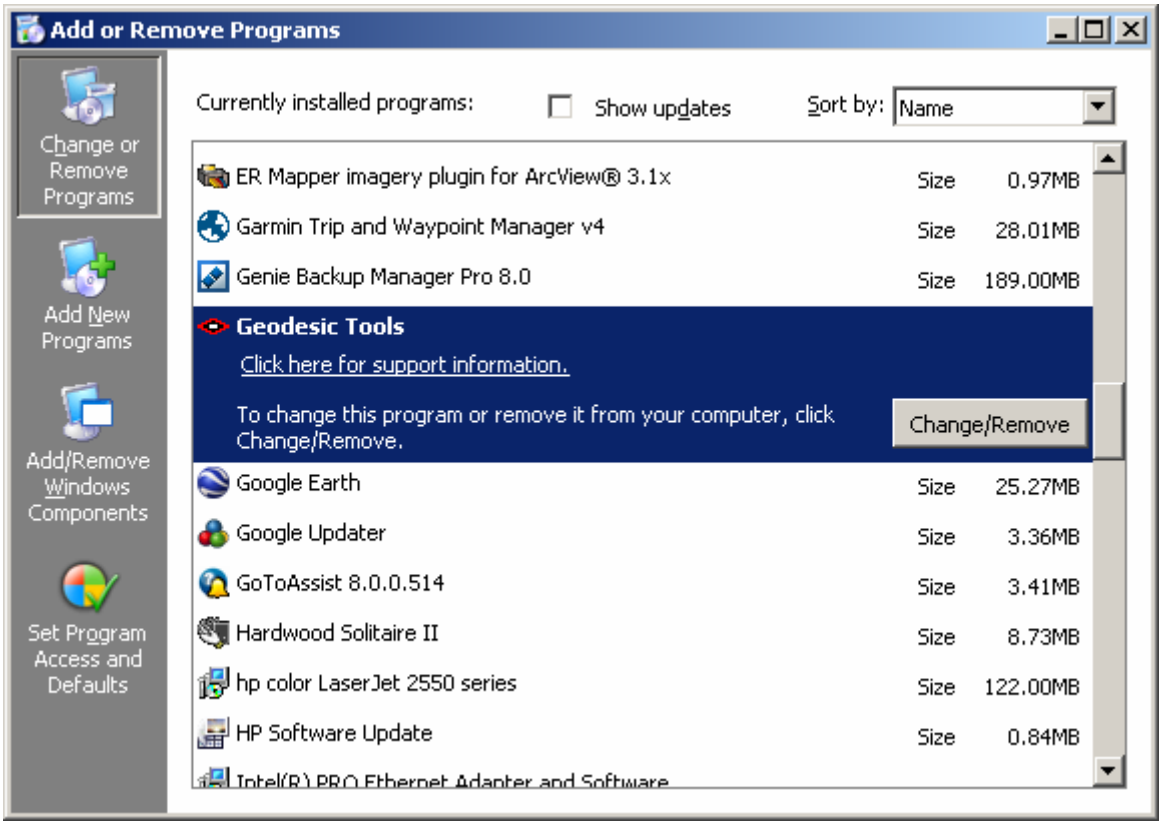
In the “Customize” dialog, click the “Toolbars” tab and check the box next to “Geodesic Tools”:



You should now see the Geodesic Tools toolbar.

Uninstalling Geodesic Tools

1) Click the Start button.
2) Open your Control Panel.
3) Double-click “Add or Remove Programs”.
4) Scroll down to find and select “Geodesic Tools”.
5) Click the “Remove” button and follow the directions.



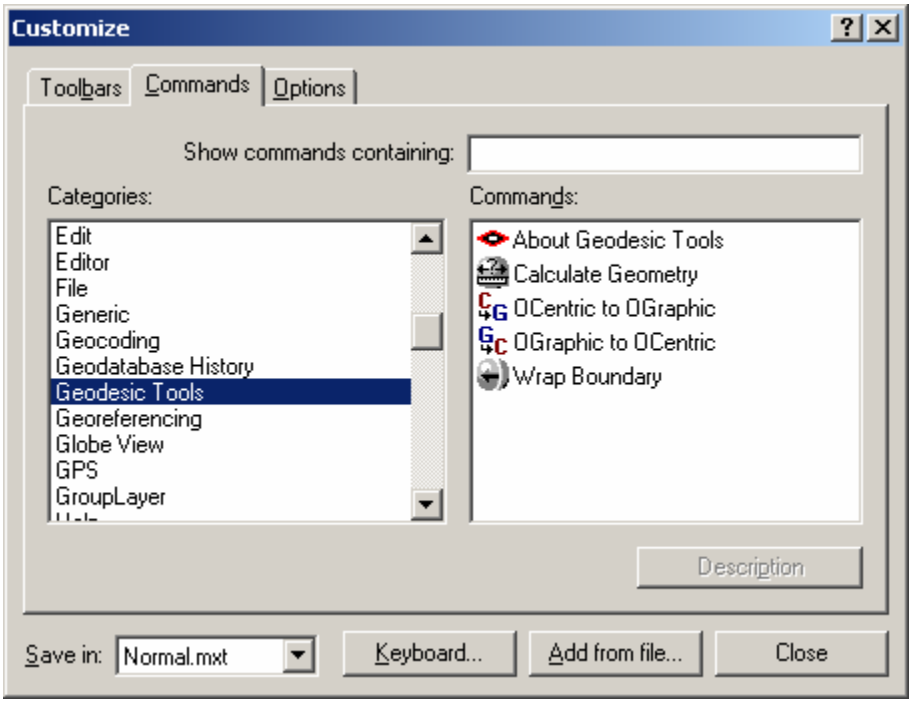
Copying and Adding Tools to Other Toolbars

Because of the way ArcGIS handles toolbars and command buttons, you may add any Geodesic Tools command buttons to any toolbar you wish. For example, if you would like to keep any of the tools available even when the Geodesic Tools menu is closed, you may easily add those tools to any of the existing ArcGIS toolbars.

To do this, open your "Customize" tool by either:

1) Double-clicking on a blank part of the ArcMap toolbar, or
2) Clicking the "Tools" menu, then "Customize".

In the "Customize" dialog, click the "Commands" tab and scroll down to select "Geodesic Tools":

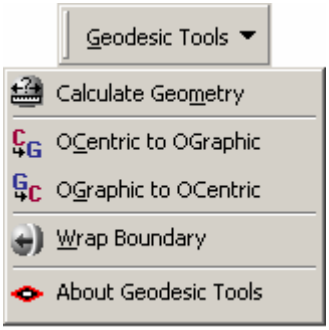


Finally, simply drag any of the commands out of the Customize dialog up into any of the existing ArcGIS toolbars.

If Any of the Tools Crash

If a tool crashes, you should see a dialog that tells us what script crashed and where it crashed. I would appreciate it if you could take screenshots of those dialogs and email them to me at jeffj@jennessent.com.

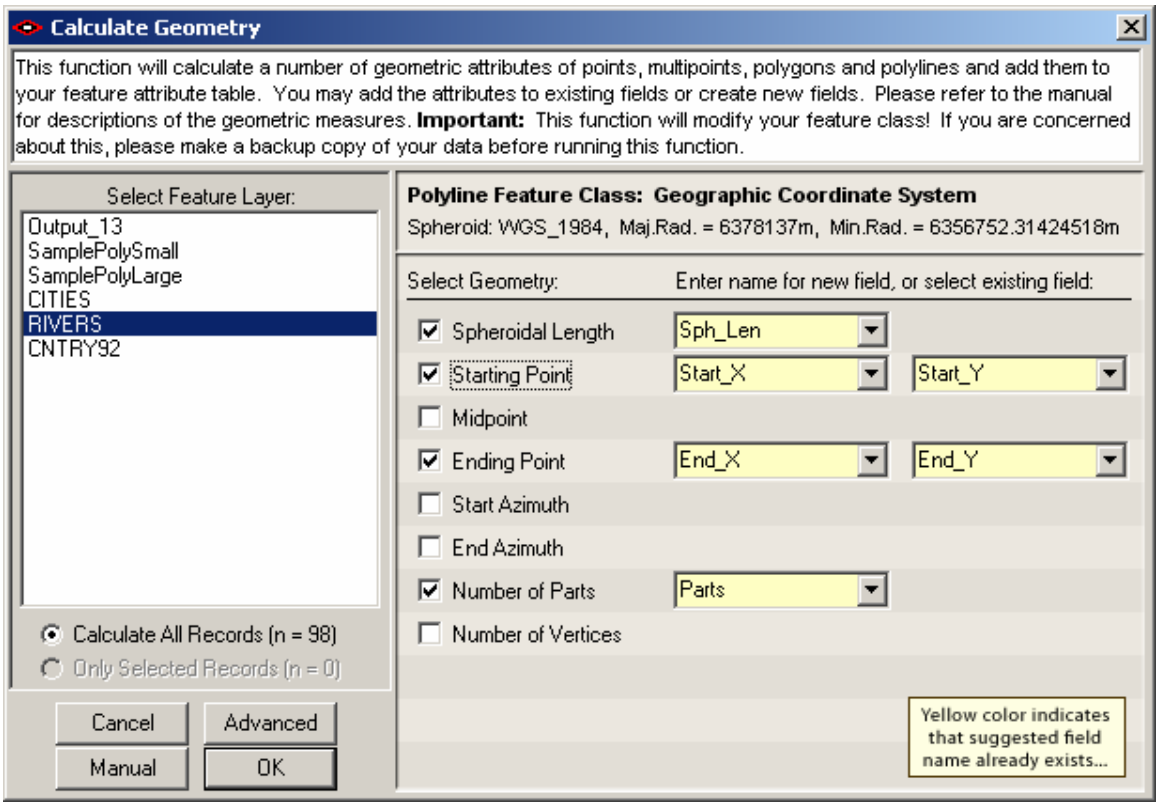
Geodesic Tools



Calculate Geometry

This function calculates a wide variety of geometric attributes of point, multipoint, polygon and polyline feature classes, including lengths, centroids and areas calculated on the sphere or spheroid. Some of these attributes may also be calculated using the standard ArcGIS "Calculate Geometry" function (i.e. open a feature class attribute table, right-click on the field name and select "Calculate Geometry"). However, this function provides many attributes that the standard ArcGIS function does not offer, and this function can add new fields to the attribute table automatically if necessary.

To run this function, click the menu item "Calculate Geometry":



The geometry options available will vary based on the projection of your feature class and what type of geometry the feature contains. See Table 1 for a full list of options for each type of feature class. Each geometry option is described in more detail after Table 1.

Note that if your feature class is projected, then you have the option to calculate both spherical and planimetric (projected) geometric attributes. The spherical attributes take longer to calculate but they are more accurate. Please refer to p. 18 for a comparison and discussion of spherical vs. projected geometric calculations.

You must specify a field name for each geometric attribute. You may either select one of the existing fields from the drop-down list or you may type a new field name in the box. If any of your specified field names already exist, they will be colored yellow in the dialog. You will be asked to confirm your choice if any of the field names already exist:

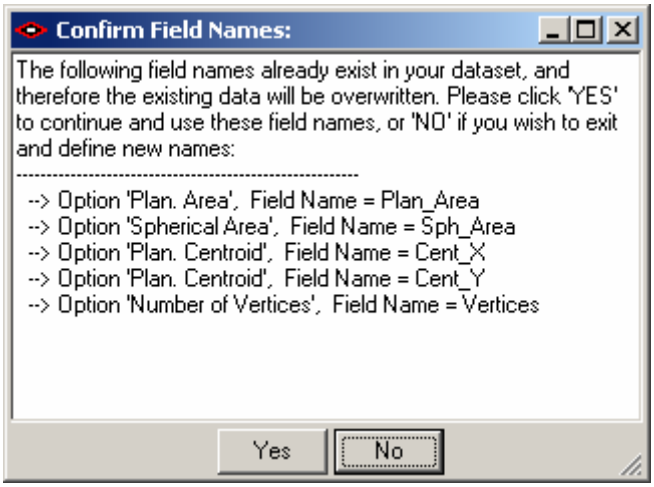


Table 1: Geometric Attributes available per feature type and projection.

Feature Type	Geometric Measure	Projection		
		Unknown	Projected	Geographic
Polygon	Planimetric Area	X	X	
	Spherical Area[1]		X[3]	X[3]
	Planimetric Centroid X	X	X	
	Planimetric Centroid Y	X	X	
	Spherical Centroid X[1]		X[4]	X
	Spherical Centroid Y[1]		X[4]	X
	Planimetric Perimeter Length	X	X	
	Spheroidal Perimeter Length[2]		X[3]	X[3]
	Number of Parts	X	X	X
	Number of Vertices	X	X	X
Polyline	Planimetric Length	X	X	
	Spheroidal Length[2]		X[3]	X[3]
	Start Point X	X	X	X
	Start Point Y	X	X	X
	Planimetric Midpoint X	X	X	
	Planimetric Midpoint Y	X	X	
	Spheroidal Midpoint X[2]		X[4]	X
	Spheroidal Midpoint Y[2]		X[4]	X
	End Point X	X	X	X
	End Point Y	X	X	X
	Projected Azimuth	X	X	
	Spheroidal Beginning Azimuth		X[4]	X[4]
	Spheroidal Ending Azimuth		X[4]	X[4]
	Number of Parts	X	X	X
	Number of Vertices	X	X	X
Point	X-Coordinate	X	X	
	Y-Coordinate	X	X	
	Latitude		X	X

	Longitude		X	X
Multipoint	Planimetric Centroid X	X	X	
	Planimetric Centroid Y	X	X	
	Spherical Centroid Latitude[1]		X[4]	X
	Spherical Centroid Longitude[1]		X[4]	X
	Number of Points	X	X	X

[1] Spherical measures based on a sphere with volume equal to selected spheroid volume (see p. 44).
[2] Spheroidal measures calculated from selected spheroid using Vincenty's equations (see p. 23).
[3] Value calculated from sphere or spheroid, but reported in meters or square meters
[4] Value calculated from sphere or spheroid, but reported in projected coordinates

NOTE: Polygon perimeters and polyline lengths and centroids are calculated on the spheroid of the original data (see p. 23). Polygon spherical areas and centroids are calculated on a sphere, regardless of whether the data projection uses a sphere or spheroid. If the data projection uses a spheroid, then the areas and centroids are calculated from a sphere containing the same volume as the original spheroid (see p. 44). Spheroids model the shape of the earth slightly better than spheres, and therefore I tried to go with the most accurate method I could find. In the case of polygon areas, however, I decided to settle for spherical rather than spheroidal methods because:

1) Even spheroids do not model the earth's surface perfectly, so there is bound to be some error regardless of the method, and
2) Topographic variation over the surface of the earth will more than wash out any marginal gains in accuracy produced by spheroidal methods, and
3) Most important of all, I simply could not find any existing formulas to help me calculate the area of a polygon over a spheroid, and I am not enough of a mathematician to easily figure them out for myself. Given (1) and (2) above, I felt that the spherical methods should provide sufficient accuracy.

Polygon Measures:

1. Planimetric Area: The area of the polygon assuming that the polygon lies on a flat surface, calculated from the coordinates of the polygon vertices. Values are reported in square map units. This is the standard area value reported by most analytical functions. The accuracy can range from very high to very low depending on the projection used. **NOTE:** Planimetric area should **not** be calculated on geographically-projected polygons! Latitude/Longitude values are polar coordinates (i.e. they reflect angular units) rather than Cartesian coordinates (i.e. X,Y units), and the methods used to calculate area from Cartesian coordinates are not appropriate for polar coordinates. A polygon area reported in "square degrees" is completely nonsensical.
2. Spherical Area: The area of the polygon assuming that it is draped over a sphere. Area values are reported in square meters. This area can only be calculated on a geographic polygon. If the original polygon is projected, it will be unprojected to latitude/longitude coordinates before the area is calculated. This area value should be very close to accurate regardless of the projection of the data, and features from different projections can therefore be compared directly without concern for projection. Please refer to p. 26 for geometric methods used in this function, and p. 18 for a comparison between spherical area and projected area.
3. Planimetric Centroid X and Y Coordinates: The coordinates of the polygon center of mass, assuming that the polygon lies on a flat surface and has vertices in Cartesian coordinates. The methods used to calculate the planimetric centroid are only appropriate if the polygon

vertices are in Cartesian coordinates (i.e. if the polygon is projected), and therefore this option is not available if the polygon is in a geographic projection (i.e. if the vertices are in polar coordinates).

4. Spherical Centroid X and Y Coordinates: The coordinates of the polygon center of mass, assuming that the polygon lies on a sphere and has vertices in polar coordinates (i.e. Latitude/Longitude values). These values can only be calculated using geographic coordinates so this option will not be offered if the projection is unknown. If the polygon is projected, then it will be unprojected to latitude/longitude values before the centroid is calculated. The centroid will then be projected back into the original polygon projection in order to get the projected X and Y Coordinates. Please refer to p. 26 for geometric methods used in this function, and p. 18 for a comparison between spherical centroids and projected centroids.
5. Planimetric Perimeter Length: The length of the polygon boundaries, calculated in the units of the map projection.
6. Spheroidal Perimeter Length: The length of the polygon boundaries as they lay over the data sphere or spheroid. Spheroidal (i.e. ellipsoidal) lengths are marginally more accurate than spherical lengths and much more accurate than projected lengths. Length values are reported in meters. Please refer to p. 23 for a description of methods used in this function.
7. Number of Parts: Some polygons are composed of multiple parts. This value simply reports the number of separate sub-polygons in each polygon feature.
8. Number of Vertices: The number of vertices in each polygon feature.

Polyline Measures:

1. Planimetric Length: The length of the polyline in the units of the map projection. This option is only appropriate for polygons in Cartesian coordinates and therefore is not available if your feature class is in a geographic projection.
2. Spheroidal Length: The length of the polyline as it truly lies on the surface of the data sphere or spheroid. Length values are reported in meters. See p. 23 for methods used in this function.
3. Start Point X and Y Coordinates: The coordinates of the polyline starting point, in the units of the polyline coordinate system.
4. Planimetric Midpoint X and Y Coordinates: The coordinates of the midpoint along the polyline, in the units of the polyline projection. The methods used to identify the planimetric midpoint are only appropriate for projected polylines, so this option is not available for geographic polylines.
5. Spheroidal Midpoint X and Y Coordinates: The coordinates of the midpoint of the polyline as it lies on the data sphere or spheroid. The methods used to identify the spheroidal midpoint are not appropriate for projected data, so projected polylines will be unprojected to latitude/longitude coordinates before this point will be calculated. This option is not available if the polyline projection is unknown. See p. 23 for methods used in this function.
6. End Point X and Y Coordinates: The coordinates of the ending point of the polyline, in the units of the polyline coordinate system.
7. Projected Azimuth: The average azimuth of the polyline, calculated as the azimuth (in degrees clockwise from North) of the straight line connecting the beginning of the polyline to the end of the polyline (see p. 36 for methods).

8. Spheroidal Starting Azimuth: The beginning azimuth of the geodesic curve connecting the beginning of the polyline to the end of the polyline. When calculating azimuth on a spheroid, the azimuth almost always changes constantly over the length of a geodesic connecting two points. This function calculates the azimuth of the geodesic as it leaves the starting point. See p. 23 for details.
9. Spheroidal Ending Azimuth: The ending azimuth of the geodesic curve connecting the beginning of the polyline to the end of the polyline. When calculating azimuth on a spheroid, the azimuth almost always changes constantly over the length of a geodesic connecting two points. This function calculates the azimuth of the geodesic as it reaches the ending point. See p. 23 for details.
10. Number of Parts: Some polylines are composed of multiple parts. This value simply reports the number of separate sub-polylines in each polyline feature.
11. Number of vertices: The number of vertices in each polygon feature.

Point Measures:

1. X and Y Coordinates: The X and Y coordinates in the Cartesian coordinate system of the feature class projection.
2. Latitude and Longitude: The polar coordinates (i.e. latitude and longitude) of the point. If the point is projected, then the point is unprojected to the projection geographic coordinate system before these values are identified.

Multipoint Measures:

1. Planimetric Centroid X and Y Coordinates: The coordinates of the multipoint center of mass, assuming the multipoint lies on a flat surface and has points in Cartesian coordinates. The methods used to calculate the planimetric centroid are only appropriate if the points are in Cartesian coordinates (i.e. if the multipoint is projected), and therefore this option is not available if the multipoint is in a geographic projection (i.e. if the points are in polar coordinates).
2. Spherical Centroid X and Y Coordinates: The coordinates of the multipoint center of mass, assuming the multipoint lies on a sphere and has points in polar coordinates (i.e. Latitude/Longitude values). These values can only be calculated using geographic coordinates, so this option will not be offered if the projection is unknown. If the multipoint is projected, then it will be unprojected to latitude/longitude coordinates before the centroid is calculated. The centroid will then be projected back into the original multipoint projection in order to get the projected X and Y Coordinates. For a review of the methods used in this function, see the discussions on calculating the polygon weighted mean centroid (p. 29) and triangle centroid (p. 37)

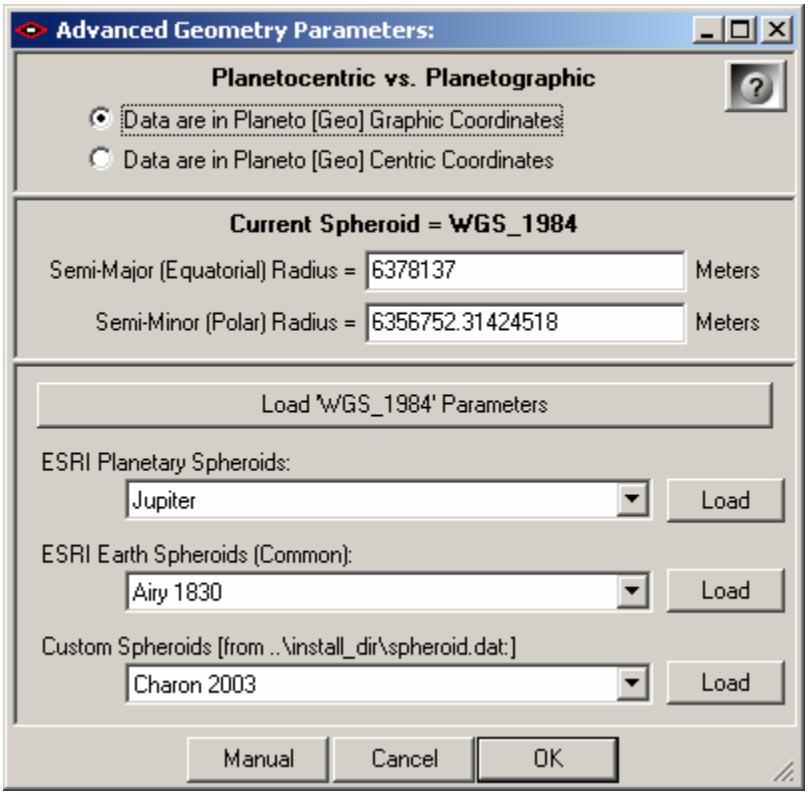
Advanced Features:

Note: The advanced features only apply to geographically-projected multipoint, polyline or polygon feature layers. The "Advanced" button will be disabled if the selected layer is not geographically projected or if it is a point layer.

Spherical and spheroidal measures are calculated on the sphere or spheroid of the data. As you select a feature layer in the "Calculate Geometry" dialog, notice that the spheroid parameters appear automatically above the list of geometry output options. In the illustration above, the "RIVERS" feature layer is seen to be in a Geographic Coordinate System based on the WGS 84 spheroid, with semi-major axis radius = 6378137m and semi-minor axis radius = 6356752.31424518m.

However, you are not forced to use this spheroid, and neither are you forced to assume that the data are in geographic (or planetographic) coordinates (see p. 45 for a comparison of planetocentric and planetographic coordinate systems).

If you wish to modify the spheroid parameters, click the "Advanced" button to open the "Advanced Geometry Parameters" dialog:



If your data are in planetocentric coordinates, then you may designate it here and the tool will adjust the analytical algorithms accordingly. You may also enter any semi-major or semi-minor radii values you wish, or enter them automatically by selecting from lists of predefined terrestrial or planetary spheroids.

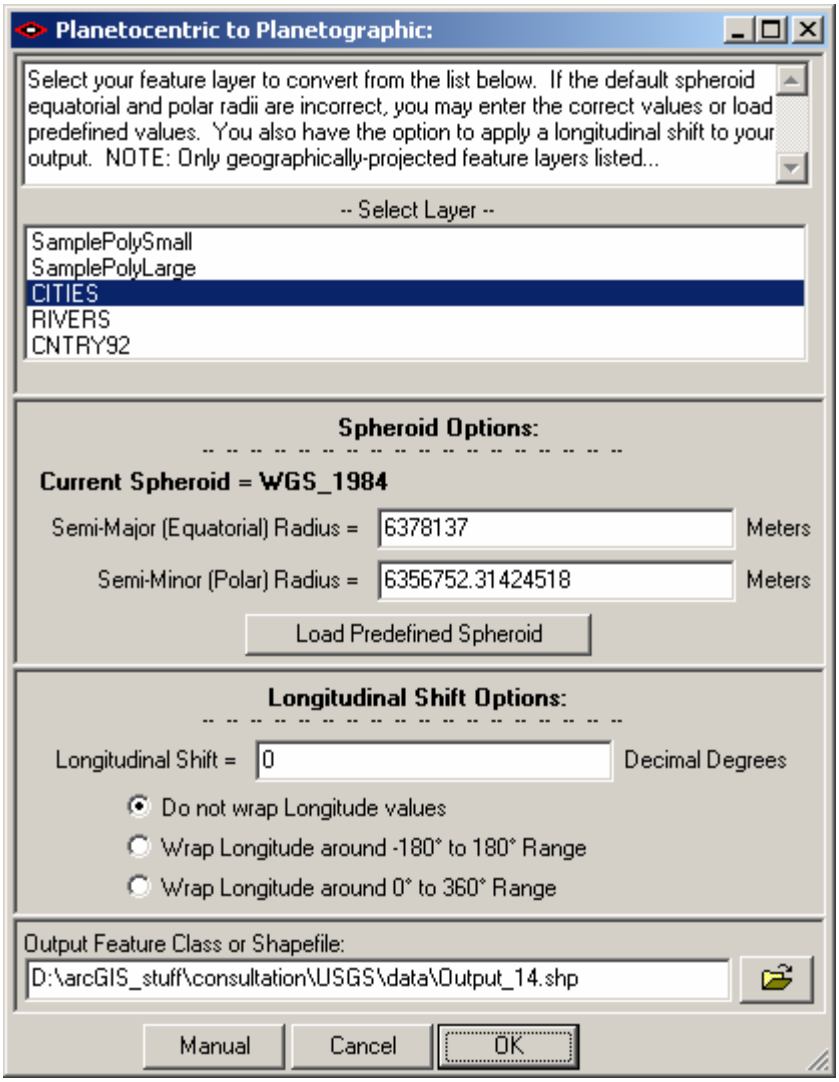
Note: If you wish to use a predefined spheroid, then select the spheroid from one of the three lists and then click the "Load" button to the right of the list. Simply selecting the named spheroid in the list will not change the current spheroid parameters.

OCentric / OGraphic Transformations

These functions transform a feature layer between planetographic and planetocentric coordinate systems, with options to specify the spheroid parameters and to apply a longitude shift. **Note:** This function only applies to geographically-projected data. You will not have the option to transform feature layers that are not in geographic coordinates.

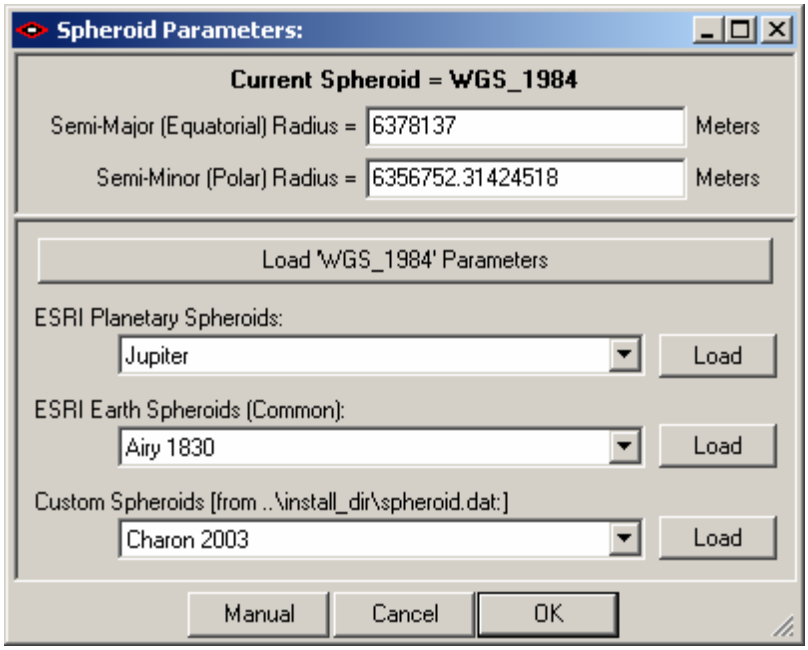
An explanation of planetocentric vs. planetographic latitudes is available on p. 45 of this manual.

Click either the "OCentric to OGraphic" or "OGraphic to OCentric" menu item to open the appropriate dialog:



Spheroid Options

The “Spheroid Options” section will automatically fill with the spheroid parameters of the currently selected feature layer. If you wish to specify a different spheroid, then click the “Load Predefined Spheroid” to open the “Spheroid Parameters” dialog:



Simply select the spheroid you prefer to use from one of the lists of pre-defined spheroids and then click the "Load" button next to that list. Alternatively, you may enter the spheroid semi-major and semi-minor radius values manually. Click 'Cancel' or 'OK' to return to the main dialog.

Longitudinal Shift Options

This section allows you to do two things:

1) Shift all features east or west by a specified number of degrees, and
2) Wrap all features around a predefined longitudinal range.

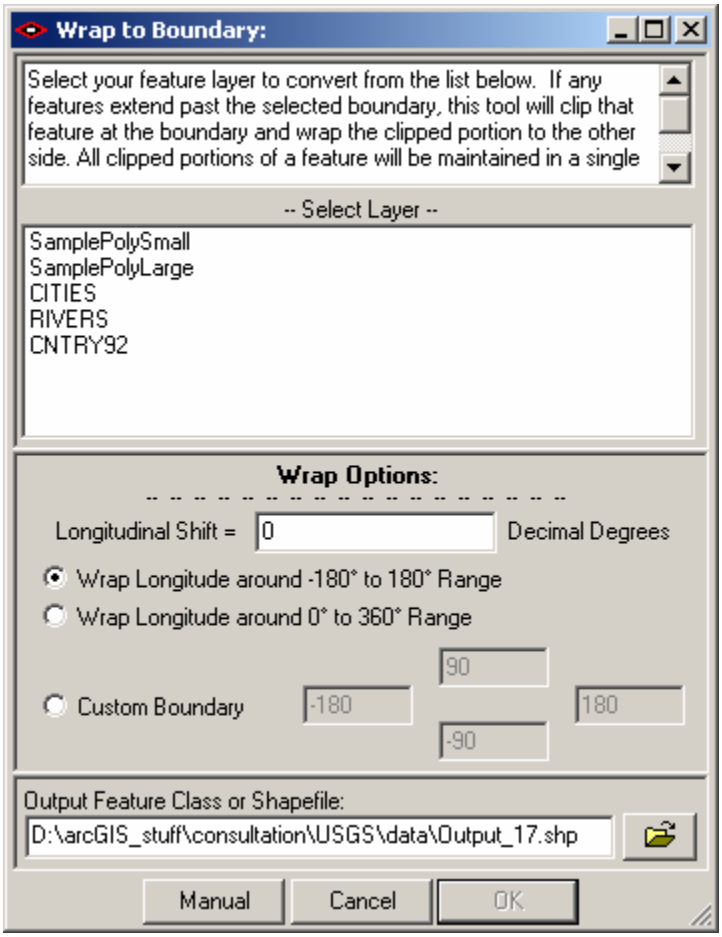
The "Longitudinal Shift" function will probably not be useful to most people working with terrestrial data. It is primarily intended for non-terrestrial datasets in which the prime meridian has been redefined. For example, if new observations lead us to redefine the coordinate system of some planet or satellite, then we might need to transform our existing datasets to account for a new prime meridian.

The "Wrap Longitude" function might be more generally useful. It forces the current dataset to fit within a pre-defined range of longitude values. If a feature extends past the eastern or western boundary, then that feature will be clipped and wrapped to the other side. Features that are split this way will still be represented with a single record in the attribute table.

Note: If you wish to shift or wrap the longitude values without also transforming between planetographic and planetocentric coordinate systems, then you may do so using the "Wrap Boundary" tool (see p. 16)

Wrap Boundary

This function allows you to shift features longitudinally and wrap them to a predefined longitudinal or latitudinal range. Click the "Wrap Boundary" menu item to open the dialog:



The “Longitudinal Shift” function will probably not be useful to most people working with terrestrial data. It is primarily intended for non-terrestrial datasets in which the prime meridian has been redefined. For example, if new observations lead us to redefine the coordinate system of some planet or satellite, then we might need to transform our existing datasets to account for a new prime meridian.

The “Wrap Longitude” function forces the current dataset to fit within a pre-defined range of longitude values. If a feature extends past the eastern or western boundary, then that feature will be clipped and wrapped to the other side. Features that are split this way will still be represented with a single record in the attribute table.

An Examination of Errors Derived from Projected Data

I was curious to see how much difference it really made to calculate the area and centroid from projected data vs. geographic data. I used the polygon feature class of United States Counties (ESRI 2006) that comes with ArcGIS as a test dataset. This dataset includes 3,219 high-resolution polygons (some with more than 60,000 vertices) delineating county boundaries in the United States and Puerto Rico.

I projected these polygons into 3 common projections (UTM, Albers Equal Area Conic and Lambert Conformal Conic; See Snyder 1983; Iliffe 2000; and Kennedy & Kopp 2000 for thorough descriptions of these map projections) and calculated the area (projected and spherical) and centroid (projected and spherical) for each projection (see p. 26 for methods). All spherical and spheroidal calculations were based on the WGS 84 spheroid. I then compared the projected vs. spherical area values for each polygon, and also the distance between the projected and spherical centroids. I assume that the spherical values are the closest to truth and I describe the difference observed in the projected values as "error".

I compared areas derived by spherical vs. projected methods by calculating the percent difference between projected and spherical area values:

$$\%\ \text{Difference} = \frac{\text{Projected Area} - \text{Spherical (True) Area}}{\text{Spherical (True) Area}}$$

A value of 50% would mean that the projected area was 1.5 × as large as the spherical area, while a value of -50% would mean the projected area was half as large as the spherical area.

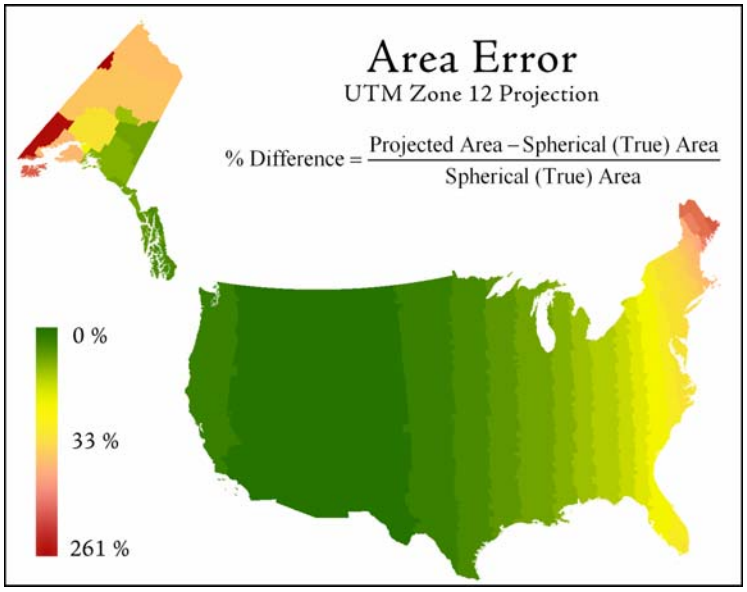
I measured the distance between spherical and projected centroids using Vincenty's equations to calculate true distances between points over the spheroid (see p. 23).

Results for UTM Zone 12

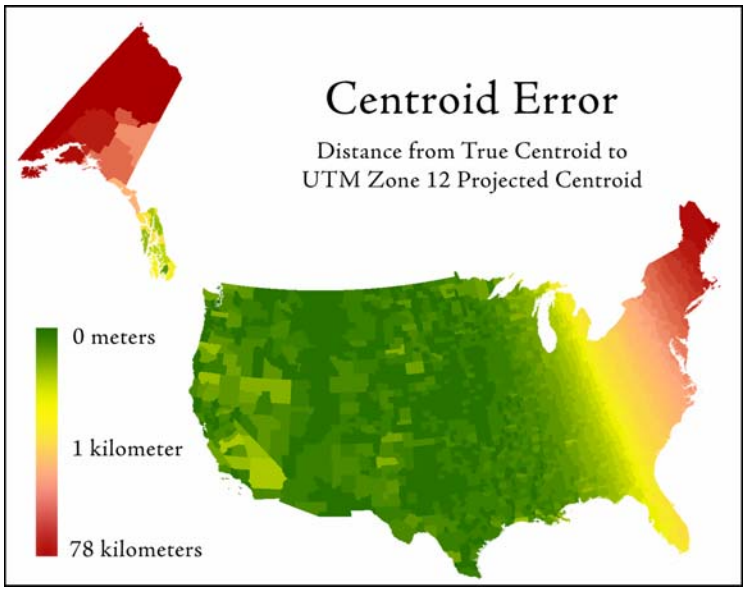
The Universal Transverse Mercator (UTM) projection is a cylindrical projection with lines of tangency along lines of longitude (hence "transverse" Mercator; the normal Mercator projection is a cylinder that is tangent on or around the equator). The UTM projection is divided into 60 zones, such that each zone has a different central meridian, and each central meridian is separated by 6º of longitude. The UTM cylinder actually intersects the planet surface along 2 longitudinal lines of tangency at ±3º east and west of the central meridian. The UTM projection is most accurate along these two lines of tangency. The central meridian for UTM Zone 12 is at Longitude -111º.

UTM is a conformal projection so it does well at maintaining the shapes of polygons. UTM Zone 12 is most accurate along the lines of Longitude -108º and -114º, and accuracy degrades as distance increases.

Not surprisingly, the errors were fairly small within the 6º band of the UTM zone. My home county (Coconino County in Arizona), located in the center of UTM Zone 12 at around 36º Latitude, has a true area of 48,272 km^2 and a projected area of 48,303 km^2, a difference of only 0.06%. Errors increase as we move east or west (see figure below), and error rates increase to approximately 0.3 – 0.6% within 300 km, and to around 1% at 600 km. Error rates increase quickly after that.



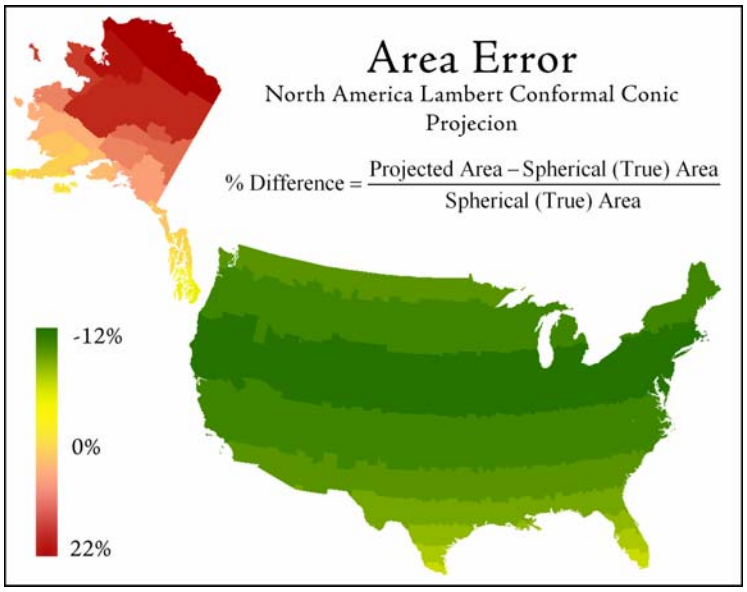
Centroid errors follow a similar pattern. These errors are affected by both the distance from the central meridian and the size of the county; smaller counties will naturally have smaller errors than larger counties that are located the same distance from the central meridian. Coconino county, which is one of the largest counties in the country, had a centroid location error of 24m, while counties ¼ the size 600km to the east had centroid locations errors in similar ranges. San Bernardino and Riverside counties, both large counties located about 150 km to the southwest of Coconino County, had centroid errors of around 180m.



Results for North America Lambert Conformal Conic Projection

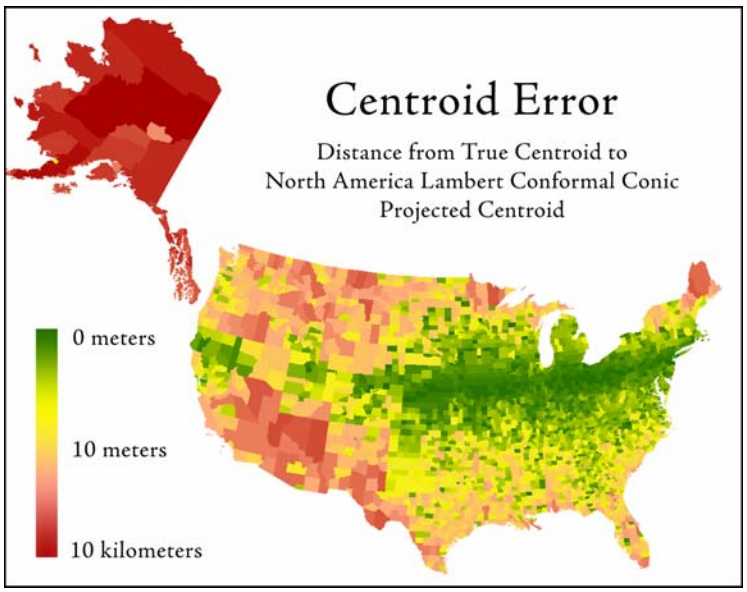
The Lambert Conformal Conic Projection is (not surprisingly) a conic projection with a latitude of origin and 2 standard parallels where the cone intersects the planetary surface (i.e. lines of tangency). The "North American" version has a latitude of origin at 40º, with lines of tangency at 20º and 60º. Therefore all distortions will be minimized along the 20º and 60º parallels of latitude. Area values tend to be compressed within the parallels of latitude and increased outside this range.

Coconino county is located fairly close to the latitude of origin and therefore the area was underestimated fairly dramatically (-11%, or over 5,300 km^2). Within the boundaries of the United States, Area values were most accurate in Hawaii (at around 20º latitude) and southern Alaska (at around 60º latitude).



As a "conformal" projection, it is intended to preserve shape as well as possible. Centroid location accuracy is associated with shape accuracy, so centroid accuracy decreases with increasing distance from the latitude of origin. Centroid accuracy also tends to decrease as the size of the polygon increases.

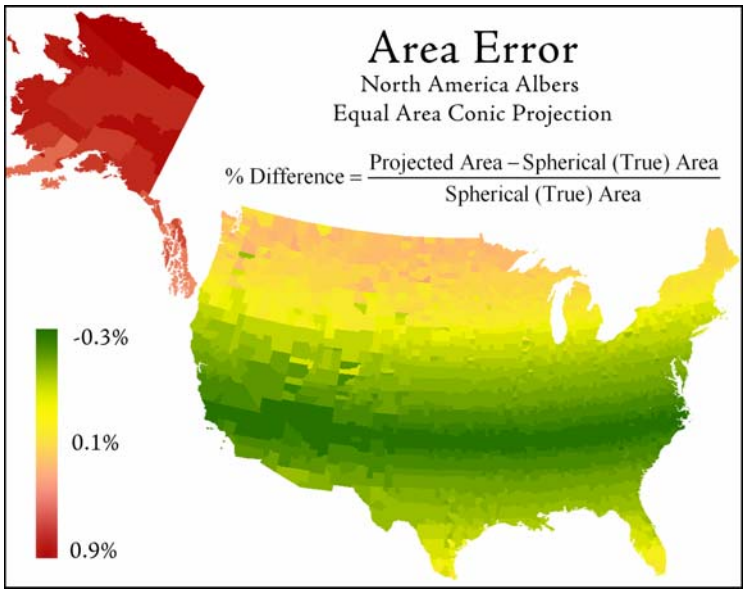
Coconino county had a centroid error of 147m, which was relatively large compared to many counties it the same latitude. For example, the 20 counties in the neighboring state of New Mexico that share the same general latitudinal range as Coconino county had an average centroid error of 14m. These 20 counties also have an average size only $\frac{1}{6}^{th}$ the size of Coconino County, which likely accounts for the difference.



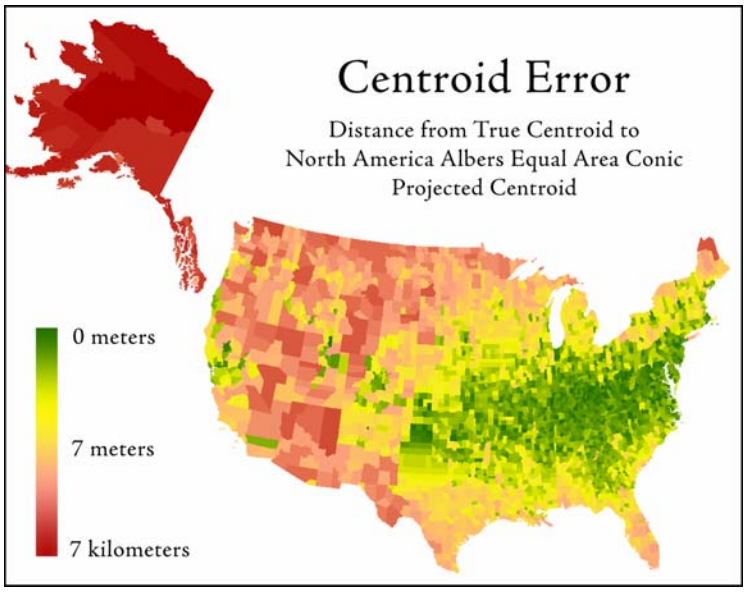
Results from North American Albers Equal Area Conic Projection

Like the Lambert Conformal Conic projection, the Albers Equal Area Conic Projection is a conic projection with a latitude of origin and two standard parallels where the cone intersects the planetary surface. It differs from Lambert in that it is designed to maintain area rather than shape (hence the name "Equal Area") and therefore the spacing between the parallels decreases toward the poles. The spacing between parallels in the Lambert projection increases toward the poles.

The North American version of Albers Equal Area has a latitude of origin at 40º and standard parallels at 20º and 60º. This projection did much better than both the UTM and Lambert projections at producing the true area of the county polygons. Accuracy does drop off with increasing distance from the latitude of origin, but the drop off is slow. The error for Coconino county, just south of 40º latitude, was only 0.01% (6 km^2).



The centroid errors were better than with the Lambert projection but worse than the UTM. Coconino county had a centroid error of 47 m while the 20 New Mexico counties at the same latitude range had an average error of 5 m.



In sum, projected datasets do have errors, and the degree of error varies dramatically depending on the projection and your location. In many cases we can get satisfactory results with projected data by choosing the right projection, but I suspect that many times people understand very little about the actual projection they are working with and they just choose whatever the default is. This can cause ridiculous errors when, for example, they try to calculate areas and distances on geographic data while treating the geographic coordinates as Cartesian coordinates (i.e. when someone calculates the area of the continental United States to be “817”, or the distance from Los Angeles to New York to be “45”).

Hopefully these tools for calculate areas and distances directly on the sphere or spheroid will help alleviate some of the confusion and provide more accurate data in general.

General Geometric Functions

Vincenty's equations for Calculations on the Spheroid

Thaddeus Vincenty, among many other ground-breaking achievements (Chovitz 2002) wrote some of the primary methods to calculate distances over a spheroid and to determine the location of a new point on a spheroid given an origin and initial bearing. He used an iterative approach in both cases to narrow down the error to an acceptable level. The equations below are numbered according to Vincenty's 1975 paper, with some modifications by Veness (2007):

Terms used by Vincenty:

- a, b = major and minor semiaxes of the ellipsoid
- f = flattening $\dfrac{a-b}{a}$
- ϕ_1 = geodetic latitude, positive north of the equator, of first point P_1
- ϕ_2 = geodetic latitude, positive north of the equator, of second point P_2
- L = difference in longitude, positive east
- s = length of geodesic $\left(\text{i.e. distance between } P_1 \text{ and } P_2 \text{ in units of } a, b\right)$
- α_1 = initial azimuth of the geodesic connecting P_1 to P_2
- α_2 = ending azimuth of the geodesic connecting P_1 to P_2
- α = azimuth of the geodesic at the equator
- $u^2 = \dfrac{\cos^2\alpha\left(a^2-b^2\right)}{b^2}$
- $U_1 = \arctan\left(1-f\right)\tan\phi_1 \quad \left(\text{i.e. reduced latitude of } \phi_1\right)$
- $U_2 = \arctan\left(1-f\right)\tan\phi_2 \quad \left(\text{i.e. reduced latitude of } \phi_2\right)$
- σ = angular distance between P_1 and P_2 on the sphere
- σ_1 = angular distance on the sphere from the equator to P_1
- σ_m = angular distance on the sphere from the equator to midpoint on the line
- λ = difference in longitude on an auxiliary sphere
- λ' = iterated estimate of λ, beginning at 2π $\left(\text{Veness 2007}\right)$

Using Vincenty's Equations to Calculate Distance and Azimuths on a Spheroid

Vincenty gives both "Direct" and "Inverse" formulae. The "Inverse" formula is used to calculate the distance between two points on the spheroid, and the initial and final azimuths of the geodesic curve connecting those two points. Numbers on the right margin refer to Vincenty's equation numbers. **NOTE:** This extension automatically uses the spheroid of the actual data in all spheroid-based calculations.

Do the following (**NOTE:** steps below include some minor algebraic modifications by Chris Veness which simplify the coding a bit):

Initially set $\lambda' = 2\pi$

while $\left|\lambda - \lambda'\right| > 10^{-12}$ $\left(\text{Threshold suggested by Veness [2007]; } \cong 0.006 \text{ mm at equator}\right)$

$$\sin\sigma = \sqrt{\left(\cos U_2 \sin\lambda\right)^2 + \left(\cos U_1 \sin U_2 - \sin U_1 \cos U_2 \cos\lambda\right)^2} \qquad [14]$$

$$\cos\sigma = \sin U_1 \sin U_2 + \cos U_1 \cos U_2 \cos\lambda \quad [15]$$

$$\sigma = \arctan[2]\left(\sin\sigma,\ \cos\sigma\right) \quad [16]^{i}$$

$$\sin\alpha = \frac{\cos U_1 \cos U_2 \sin\lambda}{\sin\sigma} \quad [17]$$

$$\cos^2\alpha = 1 - \sin^2\alpha \qquad \left(\text{Trig identity; included by Veness [2007]}\right)$$

$$\cos\left(2\sigma_m\right) = \cos\sigma - \frac{2\sin U_1 \sin U_2}{\cos^2\alpha} \quad [18]$$

$$C = \frac{f}{16}\cos^2\alpha\left[4 + f\left(4 - 3\cos^2\alpha\right)\right] \quad [10]$$

$$\lambda' = \lambda \qquad \left(\text{Introduced by Veness [2007]}\right)$$

$$\lambda = L + \left(1 - C\right) f \sin\alpha\left\{\sigma + C\sin\sigma\left[\cos\left(2\sigma_m\right) + C\cos\sigma\left(-1 + 2\cos^2\left(2\sigma_m\right)\right)\right]\right\} \quad [11]^{ii}$$

Loop[iii] until $\left|\lambda - \lambda'\right| \le 10^{-12}$

$$A = 1 + \frac{u^2}{16384}\left\{4096 + u^2\left[-768 + u^2\left(320 - 175u^2\right)\right]\right\} \quad [3]$$

$$B = \frac{u^2}{1024}\left\{256 + u^2\left[-128 + u^2\left(74 - 47u^2\right)\right]\right\} \quad [4]$$

$$\Delta\sigma = B\sin\sigma\left\{\cos\left(2\sigma_m\right) + \tfrac{1}{4}B\left[\cos\sigma\left(-1 + 2\cos^2\left(2\sigma_m\right)\right) - \tfrac{1}{6}B\cos\left(2\sigma_m\right)\left(-3 + 4\sin^2\sigma\right)\left(-3 + 4\cos^2\left(2\sigma_m\right)\right)\right\}\right] \quad [6]$$

Final Distance $s = bA\left(\sigma - \Delta\sigma\right)$ [19]

Initial Azimuth[iv] $\alpha_1 = \arctan[2]\left(\cos U_2 \sin\lambda,\ \cos U_1 \sin U_2 - \sin U_1 \cos U_2 \cos\lambda\right)$ [20][v]

Final Azimuth[iv] $\alpha_2 = \arctan[2]\left(\cos U_1 \sin\lambda,\ -\sin U_1 \cos U_2 + \cos U_1 \sin U_2 \cos\lambda\right)$ [21][v]

Using Vincenty's Equations to Calculate the Position of a Point on the Spheroid

Vincenty gives both "Direct" and "Inverse" formulae. The "Direct" formula calculates the position of the new point on the spheroid given an initial point, bearing and distance. Numbers on the right margin refer to Vincenty's equation numbers.

$$\tan U_1 = \left(1 - f\right)\tan\phi_1$$

$$\cos U_1 = \frac{1}{\sqrt{1 + \tan^2 U_1}} \qquad \left(\text{Trig identity; included by Veness [2007]}\right)$$

$$\sin U_1 = \tan U_1 \cos U_1 \qquad \left(\text{Trig identity; included by Veness [2007]}\right)$$

$$\sigma_1 = \arctan[2]\left(\tan U_1,\ \cos\alpha_1\right) \quad [1]^{v}$$

$$\sin\alpha = \cos U_1 \sin\alpha_1 \quad [2]$$

$$\cos^2\alpha = 1 - \sin_2\alpha \qquad \left(\text{Trig identity; included by Veness [2007]}\right)$$

---

i Arctan[2] function described on p. 43.

ii Algebraic modification by Veness (2007) from original Vincenty (1975) equation.

iii Veness suggests setting a maximum number of iterations possible because (as Vincenty points out) the formulas could go into an infinite loop if the two points are nearly antipodal.

iv Azimuths are in radians. See p. 43 for information on converting back to degrees.

v Arctan[2] function described on p. 43.

$$A = 1 + \frac{u^2}{16384}\left\{4096 + u^2\left[-768 + u^2\left(320 - 175u^2\right)\right]\right\} \quad [3]$$

$$B = \frac{u^2}{1024}\left\{256 + u^2\left[-128 + u^2\left(74 - 47u^2\right)\right]\right\} \quad [4]$$

$$\sigma = \frac{s}{bA} \qquad \left(\text{First approximation}\right)$$

Initially set $\sigma' = 2\pi$

while $\left|\sigma - \sigma'\right| > 10^{-12}$ $\left(\text{Threshold suggested by Veness [2007]; } \cong 0.006 \text{ mm at equator}\right)$

$$\cos\left(2\sigma_m\right) = \cos\left(2\sigma_1 + \sigma\right) \quad [5]^{\text{vi}}$$

$$\Delta\sigma = B\sin\sigma\left\{\cos\left(2\sigma_m\right) + \tfrac{1}{4}B\left[\cos\sigma\left(-1 + 2\cos^2\left(2\sigma_m\right)\right) - \tfrac{1}{6}B\cos\left(2\sigma_m\right)\left(-3 + 4\sin^2\sigma\right)\left(-3 + 4\cos^2\left(2\sigma_m\right)\right)\right\}\right] \quad [6]$$

$$\sigma' = \sigma \qquad \left(\text{Introduced by Veness [2007]}\right)$$

$$\sigma = \frac{s}{bA} + \Delta\sigma \quad [7]$$

Loop[vii] until $\left|\sigma - \sigma'\right| \le 10^{-12}$

$$\phi_2 = \arctan[2]\left(\sin U_1\cos\sigma + \cos U_1\sin\sigma\cos\alpha_1,\ \left(1-f\right)\sqrt{\left[\sin^2\alpha + \left(\sin U_1\sin\sigma - \cos U_1\cos\sigma\cos\alpha_1\right)^2\right]}\right) \quad [8]^{\text{viii,vi}}$$

$$\lambda = \arctan[2]\left(\sin\sigma\sin\alpha_1,\ \ \cos U_1\cos\sigma - \sin U_1\sin\sigma\cos\alpha_1\right) \quad [9]^{\text{viii, vi}}$$

$$C = \frac{f}{16}\cos^2\alpha\left[4 + f\left(4 - 3\cos^2\alpha\right)\right] \quad [10]$$

$$L = \lambda - \left(1 - C\right)f\sin\alpha\left\{\sigma + C\sin\sigma\left[\cos\left(2\sigma_m\right) + C\cos\sigma\left(-1 + 2\cos^2\left(2\sigma_m\right)\right)\right]\right\} \quad [11]$$

Reverse Azimuth[ix] $\alpha_2 = \arctan[2]\left(\sin\alpha,\ \ -\sin U_1\sin\sigma + \cos U_1\cos\sigma\cos\alpha_1\right)$ [12][vi]

$P_2 = \left(\phi_2,\ \lambda_1 + L\right)$ ix

---

vi Algebraic modification by Veness (2007) from original Vincenty (1975) equation.

vii Veness suggests setting a maximum number of iterations possible because (as Vincenty points out) the formulas could go into an infinite loop if the two points are nearly antipodal.

viii Arctan[2] function described on p. 43.

ix Values are in radians. See p. 43 for information on converting back to degrees.

Calculating the Surface Area and Centroid of a Spherical Polygon

The CRC Handbook of Standard Mathematical Tables and Formulae (Zwillinger 2003:367) gives the following formula for the internal area of a polygon on a sphere:

$$A = \left(\sum_{i=1}^{n} \theta_i - (n-2)\pi \right) R^2$$

where R = radius of sphere

n = number of vertices

θ_i = internal angles of polygon in radians

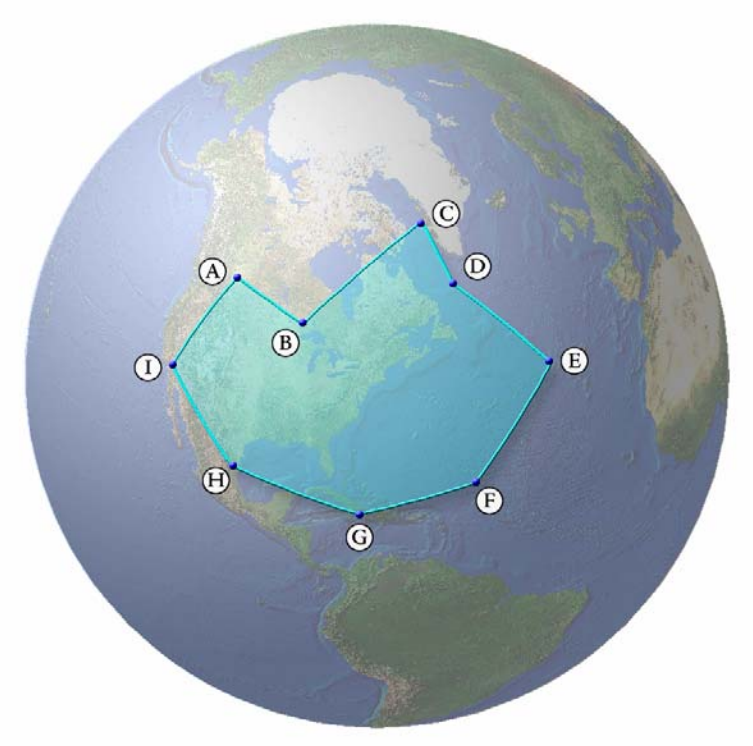
This extension does not use this method because it does not lend itself easily to multipart polygons or polygons with holes. Rather, this extension uses a slightly more complex method involving breaking the polygon down into a set of spherical triangles, and which requires approximately 33% more computation time but has the advantage of allowing us to calculate the surface centroid of that polygon at the same time.

The algorithm works as follows:

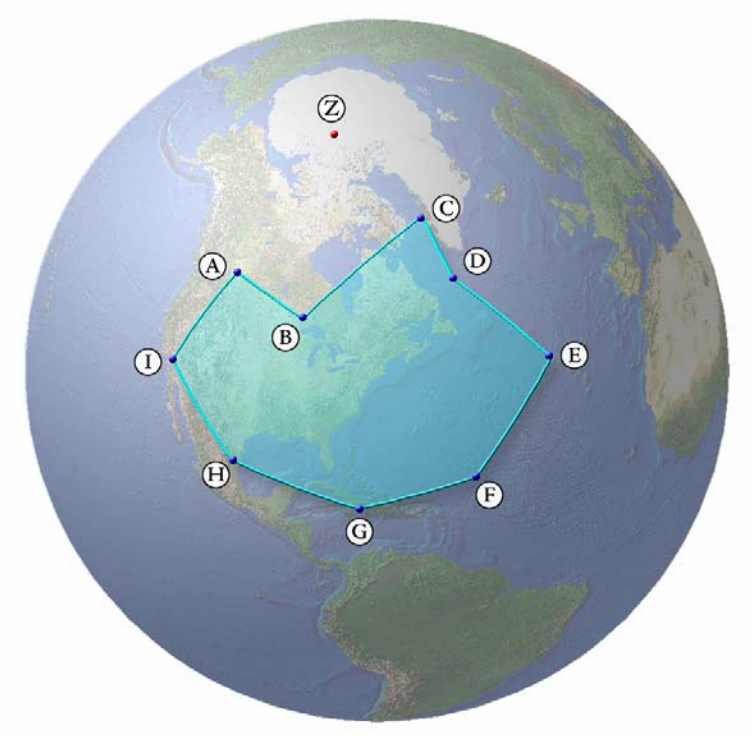
1) Break the polygon up into a set of triangles.
2) Calculate the area and center of mass of each triangle.
3) The polygon area is the sum of all the triangle areas.
4) The polygon centroid is the weighted average of all the triangle centroids, where the triangle centroid is weighted by the triangle area.

There is even an easy way to break up the polygon into triangles. We do not have to worry about generating triangles that occur only within the polygon because we can either add or subtract the triangle area depending on whether the vertices were entered clockwise or counter-clockwise. This means that triangles may be generated outside the polygon boundary, but the area of those triangles are subtracted from the cumulative area, and the final center of mass reflects only the area within the polygon boundary.

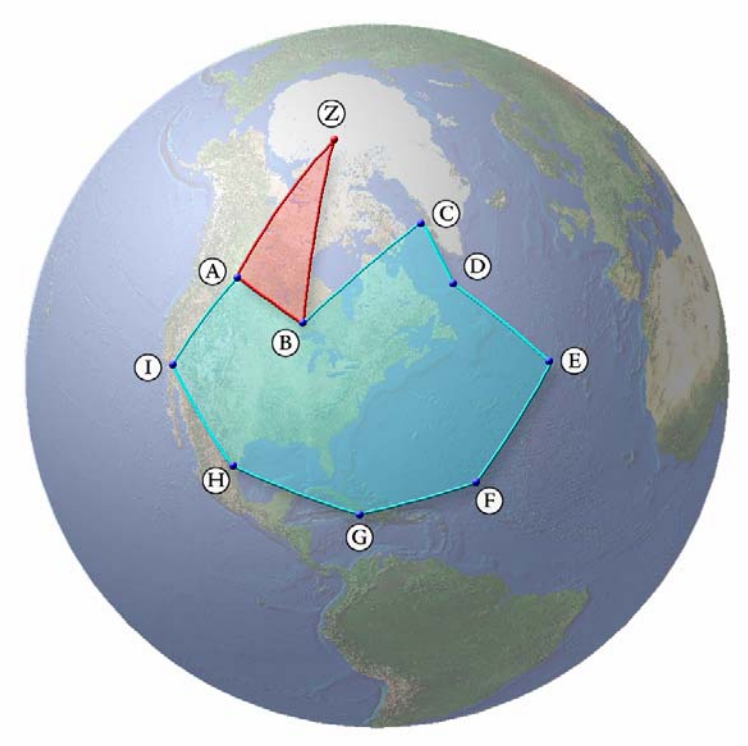
For example, if we had a polygon with 9 vertices labeled A..I:



We start by creating a new point, and then we generate triangles using that new point and all consecutive pairs of vertices. This new point does not have to be inside the polygon:

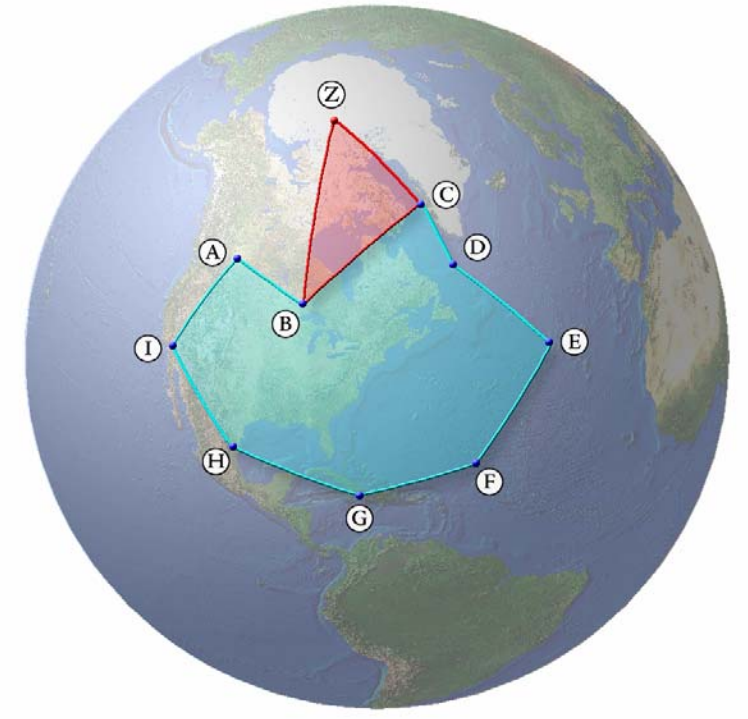


We generate triangles for each consecutive pair of vertices, such that each triangle is composed of vertices (V_i, V_{i+1}, New Point). The first triangle includes vertex A, vertex B and the New Point. Note that the vertices are entered in a counter-clockwise direction:



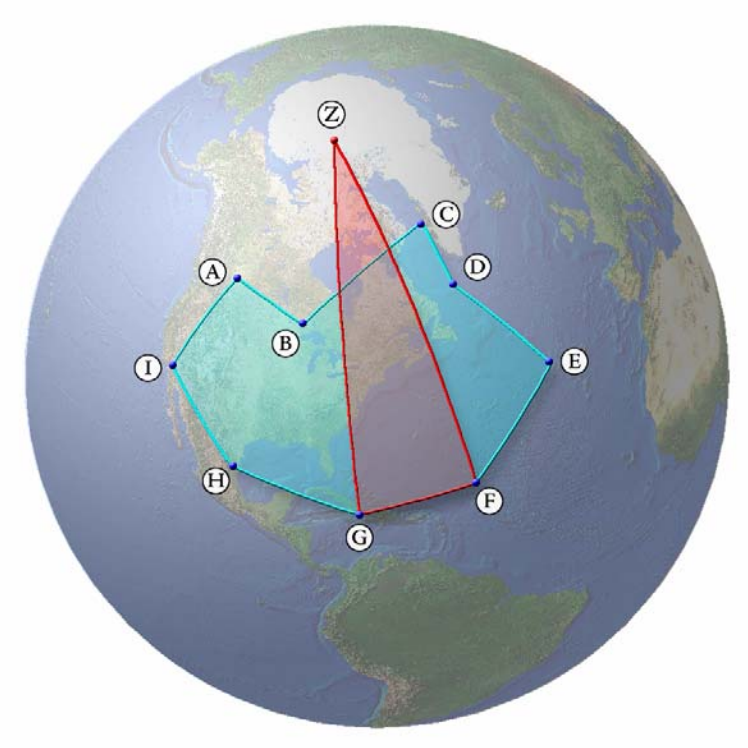
We calculate the area (p. 31) and center of mass (p. 37) of this triangle and multiply the centroid by the area to get a weighted centroid. Because these vertices are entered counter-clockwise, we set this area value to be negative.

The second triangle includes vertex B, vertex C and the New Point:



Again, these vertices are entered counter-clockwise and therefore this triangle is also assigned a negative area value.

Later on, we would generate a triangle from vertex F, vertex G and the New Point:



Note that in this case the vertices are entered in a clockwise direction. This is important because this means the sign of the area of ΔFGZ will be positive while the signs of the areas of ΔABZ and ΔBCZ were negative. The negative values of ΔABZ and ΔBCZ essentially clip out the excess area calculated from ΔFGZ . The final total will only reflect the area within the polygon. This method correctly handles holes and multipart polygons.

The final polygon centroid would be the weighted average of all the triangle centroids, calculated as the sum of the weighted centroids divided by the total polygon area. Centroids should be weighted with either positive or negative values depending on whether the vertices are clockwise or counter-clockwise.

TECHNICAL NOTE: In order to minimize the potential spherical triangle edge lengths, this extension uses the projected centroid of the polygon for the New Point (Point Z in the examples above). Theoretically any point would work, but it stands to reason that a point that minimizes the distances would be likely to introduce less rounding error into the final equations, especially when using the Haversine functions.

Spherical Polygon Centroid = Mean of all spherical triangle centroids, where each centroid is weighted by triangle area (Note that centroids are in Cartesian coordinates, not polar latitude / longitude values):

Spherical Polygon Centroid:

$$X_{SPC} = \frac{\sum X_i A_i}{\sum A_i} \qquad Y_{SPC} = \frac{\sum Y_i A_i}{\sum A_i} \qquad Z_{SPC} = \frac{\sum Z_i A_i}{\sum A_i}$$

where A_i = Area of Spherical Triangle i (might be negative value if triangle is being subtracted)

These X,Y and Z values are all in Cartesian coordinates, so this final polygon centroid vector needs to be converted back to latitude and longitude values (see p. 42)

Calculating the Surface Area of a Spherical Triangle

The surface area of a spherical triangle is surprisingly easy to calculate, provided you can determine the three internal angles. Remember that the internal angles of a spherical triangle will always be greater than 180º. If the internal angles add to exactly 180º, then that triangle exists on a plane rather than a sphere. The amount by which the internal angles exceed 180º is called the *Spherical Excess*. The surface area of the spherical triangle is directly related to the spherical excess.

$$\text{Area of Spherical Triangle} = R^2 E$$

where R = radius of sphere

E = triangle spherical excess in radians

= Sum of angles - 180 (in degrees)

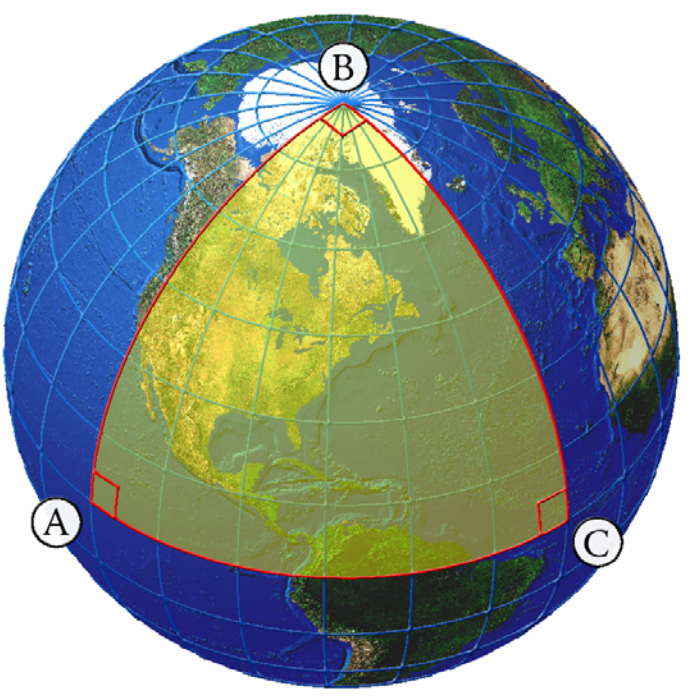
= Sum of angles - π (in radians)

For a simple example, consider the triangle formed by the 3 coordinates:

Point A: Longitude = -120, Latitude = 0

Point B: Longitude = -120, Latitude = 90

Point C: Longitude = 0, Latitude = 0



This triangle, sometimes referred to as the "quadrantal triangle", is formed by 3 right angles. The sum of the internal angles is 270º and the spherical excess is therefore 270º - 180º = 90º, or $\frac{\pi}{2}$ radians. Note that if this were the unit sphere (i.e. a sphere with radius = 1), then by our equation

$\frac{\pi}{2}$ would actually be the area of this triangle. The earth has a radius of approximately 6,371km (see p. 44), so the surface area of this large triangle = $6371^2 \frac{\pi}{2} \cong 63.8$ million km^2.

Note that this result conforms to the known value of the surface area of a sphere:

$$\text{Surface Area of Sphere} = 4\pi r^2$$
$$\text{where } r = \text{radius of sphere}$$

The triangle with 3 right angles in the example above covers exactly $\frac{1}{8}^{\text{th}}$ of the globe:

$\left(\frac{1}{8} \times 4\pi r^2\right) = \frac{\pi}{2} r^2$, exactly as we calculated above.

The hard part is calculating the internal angles of the spherical triangle. This could be done using trigonometry to calculate the bearings to and from each pair of points (see p. 35 for example). I actually find bearings on a sphere to be somewhat confusing, though, because the bearing constantly changes as you follow a great circle route (unless you are following the equator or a line of longitude), so I found it to be a little more intuitive to calculate the triangle area using the triangle edge lengths instead. The following formula (also based on the triangle spherical excess) calculates the spherical area of the triangle from edge lengths (adapted from Zwillinger 2003:370).

$$\text{Area} = \tan\left(\frac{E}{4}\right) = \sqrt{\tan\left(\frac{S}{2}\right)\tan\left(\frac{S-\overrightarrow{AB}}{2}\right)\tan\left(\frac{S-\overrightarrow{BC}}{2}\right)\tan\left(\frac{S-\overrightarrow{CA}}{2}\right)}$$

$\overrightarrow{AB}, \overrightarrow{BC}, \overrightarrow{CA} = \text{Triangle Edge Lengths}$

$$S = \frac{\overrightarrow{AB} + \overrightarrow{BC} + \overrightarrow{CA}}{2} = \text{Triangle Half-Perimeter}$$

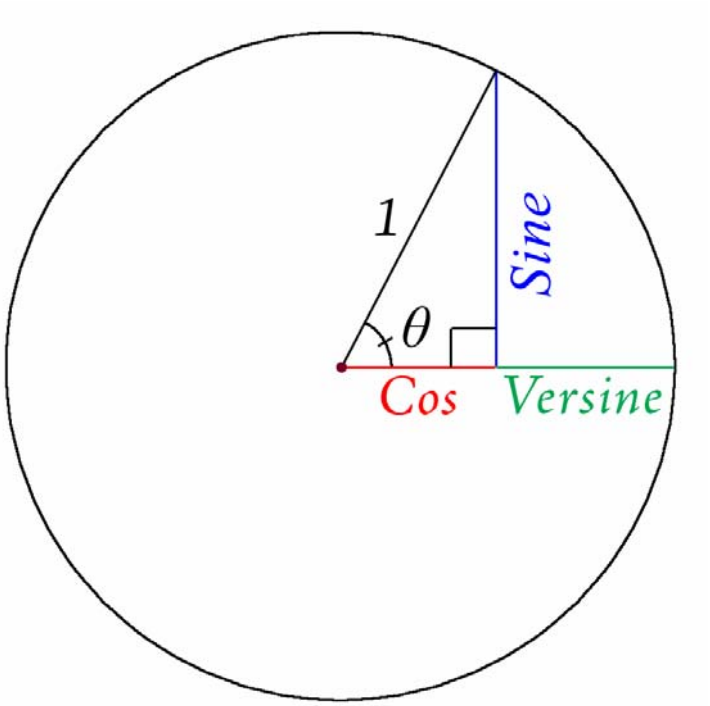
NOTE: It is worth remembering that whenever you draw a triangle on the sphere, you are actually drawing *two* triangles on the sphere. The region "outside" your triangular area of interest is also a perfectly valid triangle formed by 3 straight lines and 3 internal angles. Usually we automatically assume that the smaller of the two triangles is the one we are interested in (as the formula using edge lengths above does). However, if we are doing any function that requires calculating the internal angles of a triangle or polygon, we must take care that the angle reflects the inside of the object we are analyzing.

Calculating the Length of a Line on a Sphere

Spherical triangle edge lengths are calculated using an implementation of the Haversine formula (see Veness 2007a; Wikipedia 2007; Zwillinger 2003:373). The haversine is simply a trigonometric identity like the Sine or Cosine, although it is not used nearly as often as these more familiar identities. Haversines are useful for very small angles where cosine gets very close to 0. It has been used historically for geographic calculations over the surface of the earth because angles tend to be very small in these cases. The haversine is equal to half of the Versine, and is calculated as follows:

$$\text{Versin}(\theta) = 1 - \cos(\theta) = 2\sin^2\left(\frac{\theta}{2}\right)$$

$$\text{Haversin}(\theta) = \frac{\text{Versin}(\theta)}{2} = \sin^2\left(\frac{\theta}{2}\right)$$



The haversine is used to calculate the distance between two geographic points as follows:

$$\text{Distance} = RC$$

where R = Radius of sphere in whatever units are appropriate

$$C = 2\arctan[2]\left(\sqrt{A},\ \sqrt{1-A}\right)$$

$\arctan[2]$ = Arctangent, adjusted for quadrant (see General Geometric Functions)

Using Haversine: $A = \sin^2\left(\frac{\Delta\text{Lat}}{2}\right) + \cos\left(\text{Lat}_1\right)\cos\left(\text{Lat}_2\right)\sin^2\left(\frac{\Delta\text{Long}}{2}\right)$

$\Delta\text{Long} = \text{Long}_2 - \text{Long}_1$, in radians

$\Delta\text{Lat} = \text{Lat}_2 - \text{Lat}_1$, in radians

$\text{Lat}_1, \text{Long}_1$ = Latitude and Longitude at Point 1, in radians

$\text{Lat}_2, \text{Long}_2$ = Latitude and Longitude at Point 2, in radians

Calculating the Azimuth Between Points on the Sphere

Calculating the azimuth or bearing between two points on the sphere is conceptually much more complicated than calculating the bearing on a plane. Except in the extremely rare instance where both points lie exactly on the equator or along the same line of longitude, the bearing changes constantly as you travel over the great circle geodesic that connects those two points. For example, if you followed the great circle route from New York City, USA to Hanoi, Vietnam, you would leave New York on a bearing of 358º, travel in a straight line the entire way, and then arrive in Hanoi on a bearing of 182º.

There are various formulae available for calculating the initial bearing and the final bearing, and the bearing for points in between. The formula below calculates the initial bearing for the great circle line going from Point 1 to Point 2 (Williams 2006):

$$\text{Initial Bearing} = \arctan[2](y,\ x)$$

where $\arctan[2]$ = Arctangent, adjusted for quadrant (see General Geometric Functions)

$$y = \sin(\Delta\text{Long})\cos(\text{Lat}_2)$$

$$x = \cos(\text{Lat}_1)\sin(\text{Lat}_2) - \sin(\text{Lat}_1)\cos(\text{Lat}_2)\cos(\Delta\text{Long})$$

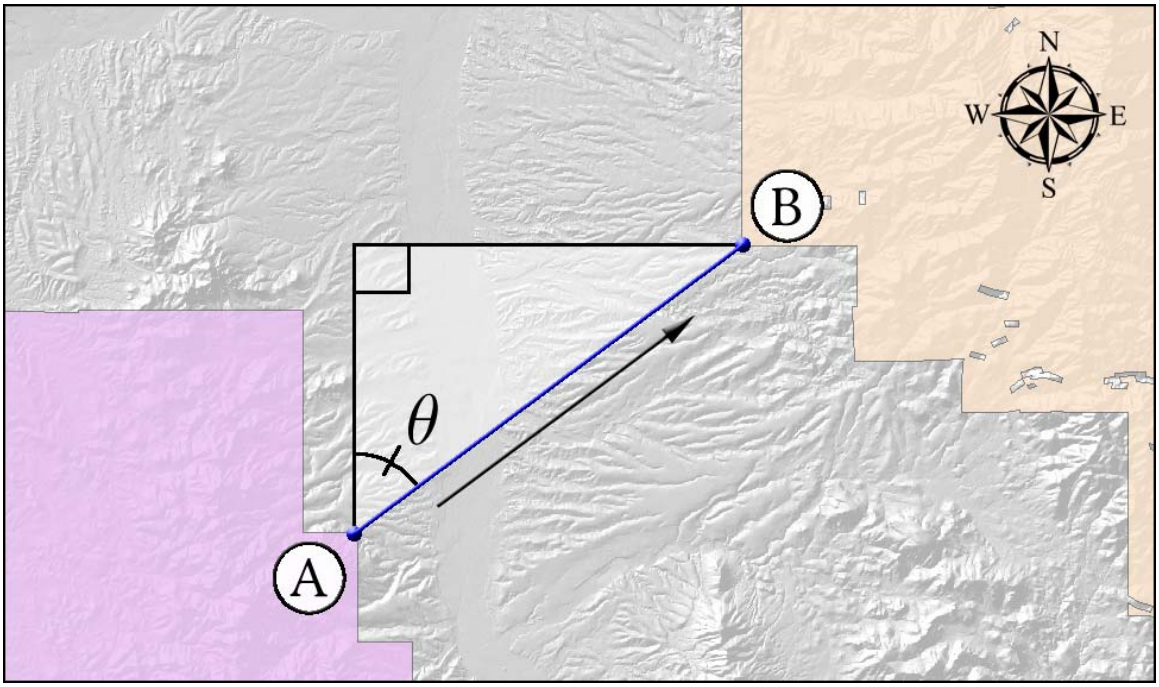
ΔLong = change in longitude from Point1 to Point2

Lat_1 = Latitude at Point 1, in radians

Lat_2 = Latitude at Point 2, in radians

Note: Result is in radians, where North = 0 and π = South

Calculating the Azimuth Between Points on a Plane



The azimuth (or bearing) between points A and B is calculated as follows:

Given that:

Δx = Point B x-coordinate – Point A x-coordinate

Δy = Point B y-coordinate – Point A y-coordinate

If If $\Delta x = 0$ and $\Delta y = 0$ then $\theta = -9999$ (i.e. No Direction)

Else if $\Delta y = 0$ then:

If $\Delta x < 0$ then $\theta = -90^{o}$

If Else If $\Delta x = 0$ then $\theta = 0^{o}$

Else If $\Delta x > 0$ then $\theta = 90^{o}$

Else:

$$\theta_a = \frac{\arctan\left(\frac{\Delta x}{\Delta y}\right) * 180}{\pi}$$

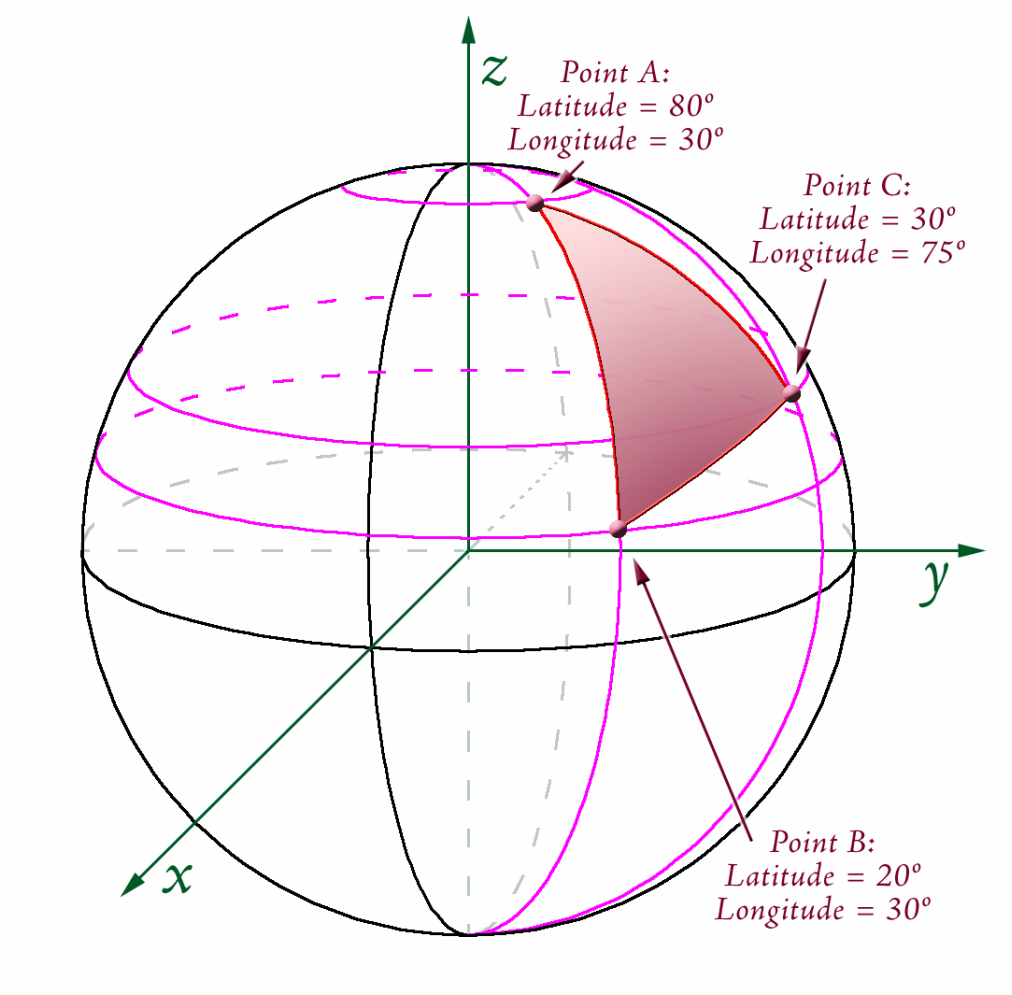
If $\Delta y \geq 0$ then $\theta = \theta_a + 180$

Else:

If $\Delta x \leq 0$ then $\theta = \theta_a$

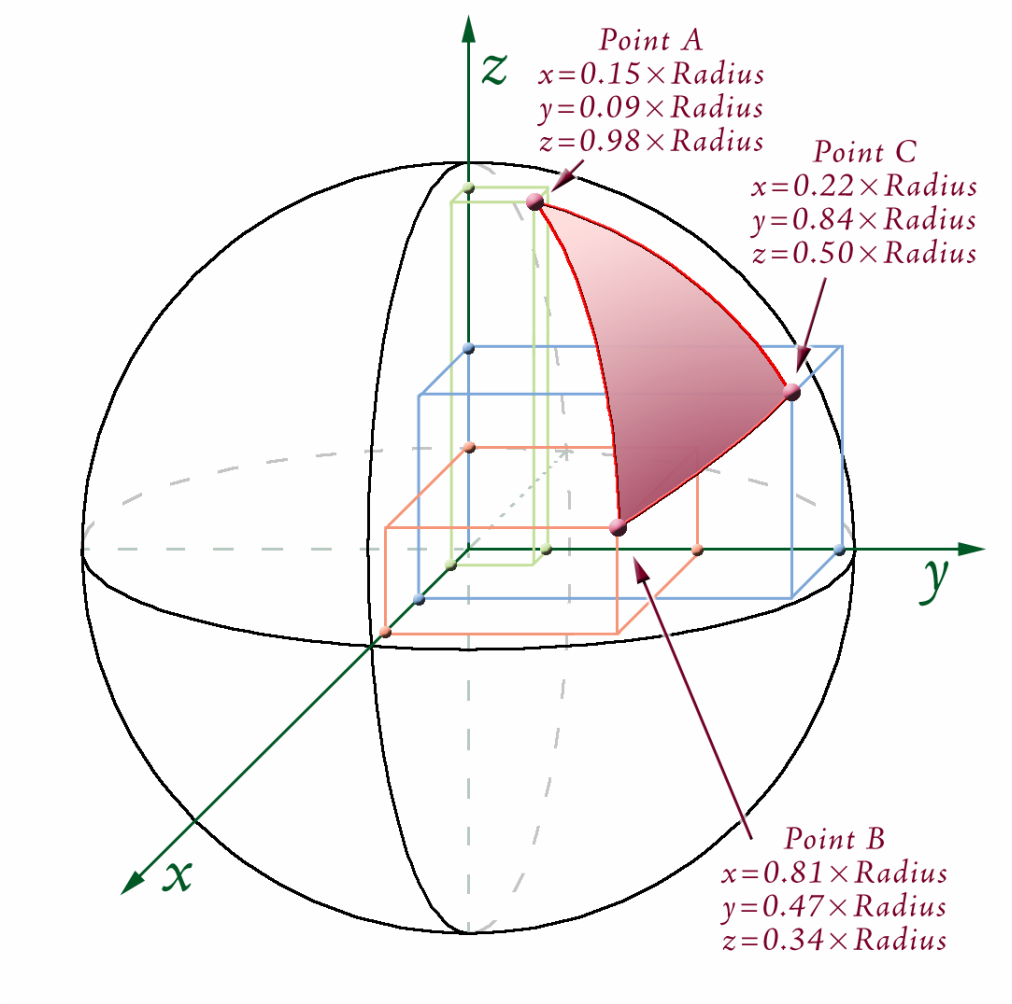
Else if $\Delta x > 0$ then $\theta = \theta_a + 360$

Calculating the Centroid of a Spherical Triangle



Calculating the centroid of a spherical triangle is similar in concept to calculating the centroid of a planar triangle. However, we must remember that longitude and latitude values are not so much coordinates as they are directions from the origin of the sphere. Therefore they cannot be added and divided the way that Cartesian coordinates can. However, they can easily be converted to Cartesian coordinates, and then the computations are simple:

Step 1: Convert Latitude and Longitude values to Cartesian coordinates (see p. 40):



Step 2: Calculate 3D Centroid of 3 points:

3D Triangle Centroid:

$$X_{3DC} = \frac{\sum(X_1 + X_2 + X_3)}{3} \qquad Y_{3DC} = \frac{\sum(Y_1 + Y_2 + Y_3)}{3} \qquad Z_{3DC} = \frac{\sum(Z_1 + Z_2 + Z_3)}{3}$$

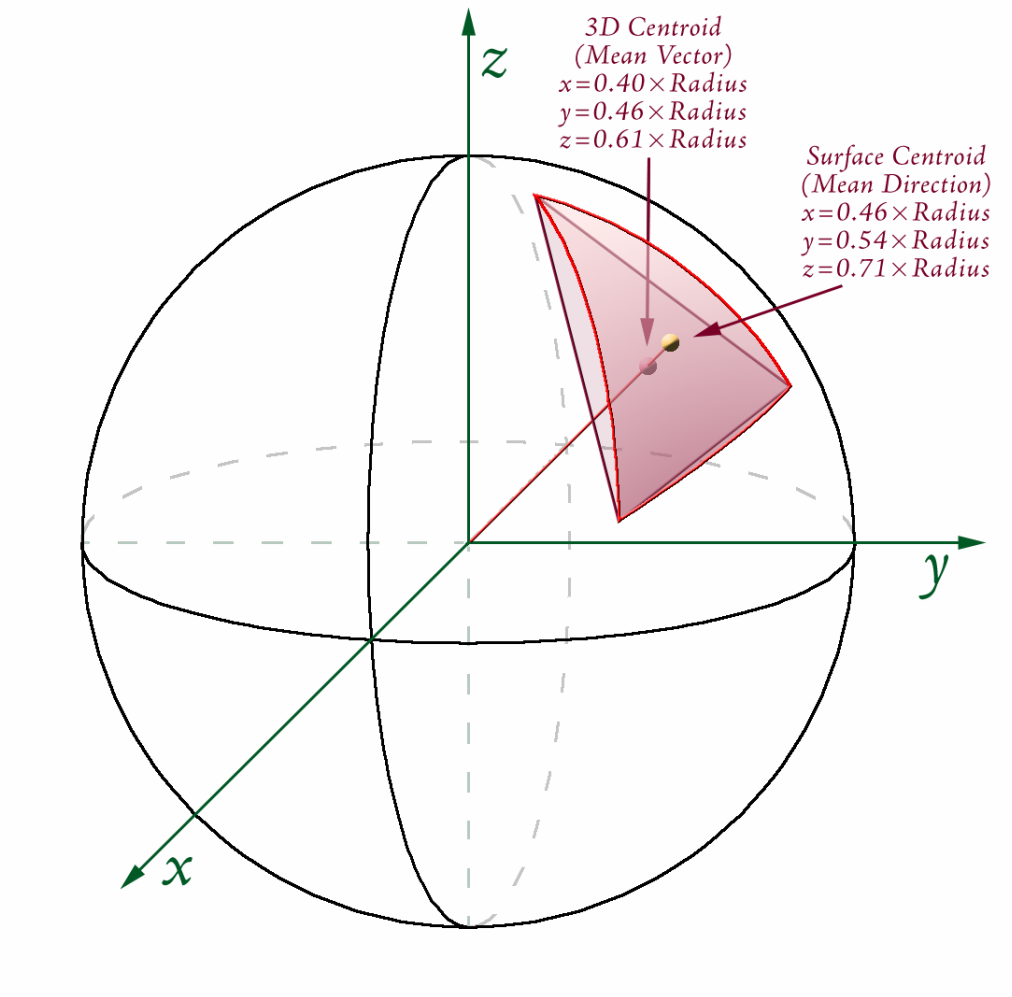
Step 3: This 3D Centroid is actually located inside planet. Shift point outward along centroid vector to the surface of the planet.

$$\text{Centroid Vector Length } = L = \sqrt{X_{3DC}{}^2 + Y_{3DC}{}^2 + Z_{3DC}{}^2}$$

$$\text{Surface X} = \frac{X_{3DC}}{L} \qquad \text{Surface Y} = \frac{Y_{3DC}}{L} \qquad \text{Surface Z} = \frac{Z_{3DC}}{L}$$

NOTE: Mardia (2000; p. 163 – 167) describes Step 2 as calculating the mean vector, and Step 3 as calculating the mean direction of the 3 vectors. The mean direction normalizes the mean vector by the mean vector length. For calculating centroids on the surface of the sphere, the mean direction is more useful than the mean vector. In our case, all points have exactly the same length, so we normalize using the sphere radius. In other words, if we consider a Lat/Long coordinate to be composed of three components (R, ϕ, θ; see p. 40 for definitions), then Step 2

identifies the mean coordinate of ϕ and θ (i.e. mean vector), while Step 3 extends this mean vector outward to the surface of the sphere by setting the vector length = R.



Converting between Latitude / Longitude and Cartesian Coordinates

Latitude / Longitude to Cartesian:

Given a Latitude and Longitude in Degrees, calculate X, Y and Z as follows:

$$\phi = \frac{(90 - \text{Latitude})\pi}{180} = \text{Angle from North Pole down to Latitude, in Radians}$$

$$\theta = \frac{(\text{Longitude})\pi}{180} = \text{Angle from Greenwich to Longitude, in Radians}$$

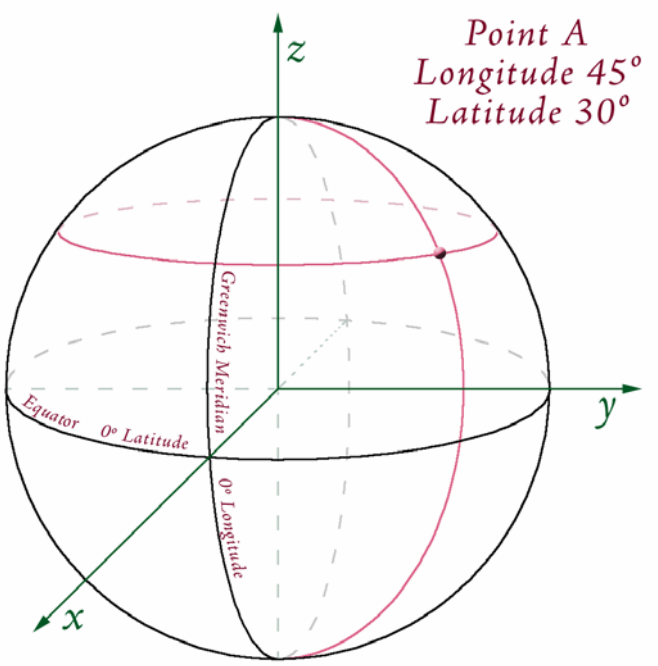
$$\text{X} = R \sin\phi \cos\theta$$

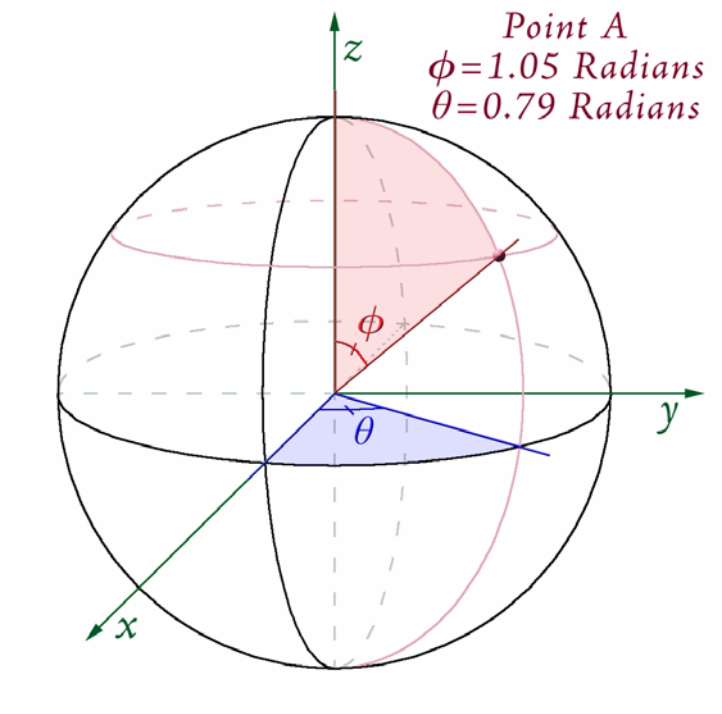
$$\text{Y} = R \sin\phi \sin\theta$$

$$\text{Z} = R \cos\phi$$

R = Radius of Sphere

For example, a point at 30º Latitude and 45º Longitude would be converted to ϕ = 1.05 radians and θ = 0.79 radians:

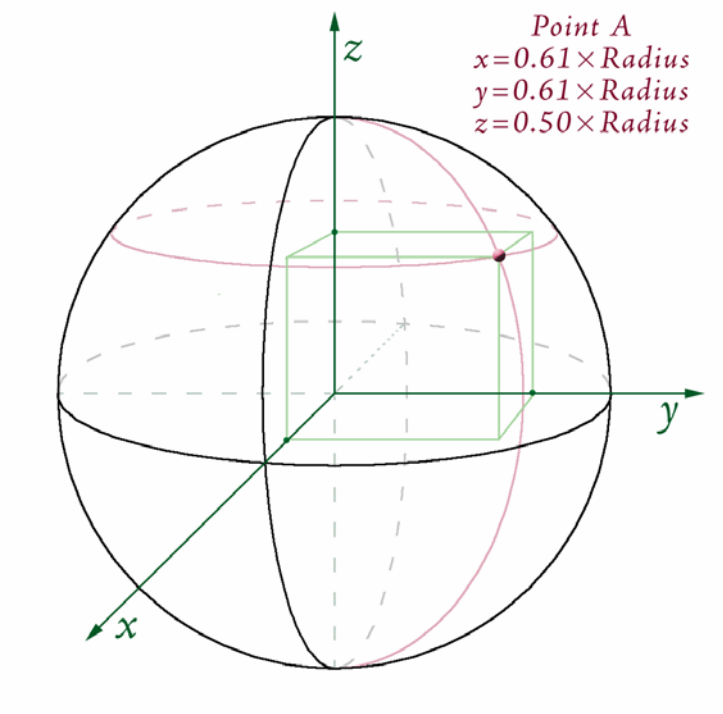




NOTE: Be careful with the terms ϕ and θ! This document treats ϕ as the angle south from the North Pole, and θ as the angle east from Greenwich. However, the literature is not consistent with these two terms! Much of the literature uses these exact two symbols, but for the other angle (i.e. ϕ for east/west and θ for north/south).

Using the formula above, calculate X, Y and Z from ϕ and θ.

$$\mathrm{X} = R\sin\phi\cos\theta = R\sin(1.05)\cos(0.79) = 0.61R$$
$$\mathrm{Y} = R\sin\phi\sin\theta = R\sin(1.05)\sin(0.79) = 0.61R$$
$$\mathrm{Z} = R\cos\phi = R\cos(0.79) = 0.5R$$



NOTE: As with ϕ and θ, the literature is inconsistent regarding how to write the X, Y and Z coordinates of a 3-dimensional point. If a point is located at X, Y, Z, some sources will write the coordinates as (X,Y,Z) while other sources will write it as (Z,X,Y). If you write coordinates in either of these formats, please identify which format you are using!

Cartesian to Latitude / Longitude:

$$\phi = \arctan[2]\left(\sqrt{X^2 + Y^2},\ Z\right) = \text{Angle from North Pole down to Latitude, in radians}$$

$$\theta = \arctan[2]\left(Y,\ X\right) = \text{Angle from Greenwich to Longitude, in radians}$$

where arctan[2] = Arctangent, adjusted for quadrant (see General Geometric Functions)

$$\text{Latitude} = 90 - \phi\left(\frac{180}{\pi}\right)$$

$$\text{Longitude} = \theta\left(\frac{180}{\pi}\right)$$

Arctan[2] Function

Various functions in this extension calculate arctangents. However, there is a problem with the basic function arctan because it does not account for quadrant. For example, given that $\tan A = \frac{\Delta Y}{\Delta X}$, then $\arctan \frac{\Delta .Y}{\Delta X} = A$, where A is in radians. However, this simple arctan function does not properly account for the signs of ΔX and ΔY and will only return values ranging between $\pm \frac{\pi}{2}$. The arctan[2] function checks the signs of ΔX and ΔY and returns a value of A radians that correctly ranges from $-\pi$ to π.

Unfortunately, Visual Basic 6 does not have a function to calculate arctangent in this manner. Many programming languages such as C++, PHP, C# and VB.NET include the "atan2" function which does the trick. You simply specify X and Y separately and the function determines the quadrant. **NOTE:** Microsoft Excel also has an "atan2" function, but for some reason the Excel version takes the ΔX and ΔY values in the order of (x, y) while all other implementations in the civilized world appear to take these values in the order of (y, x). Therefore be careful if you use this function in both your code and in Excel!

Given the lack of an Atan2 function in VB6 and VBA, I wrote my own function as follows:

```
Const dblPi As Double = 3.14159265358979
Public Function atan2(Y As Double, X As Double) As Double
  If X > 0 Then
    atan2 = Atn(Y / X)
  ElseIf X < 0 Then
    If Y = 0 Then
      atan2 = (dblPi - Atn(Abs(Y / X)))
    Else
      atan2 = Sgn(Y) * (dblPi - Atn(Abs(Y / X)))
    End If
  Else    ' IF X = 0
    If Y = 0 Then
      atan2 = 0
    Else
      atan2 = Sgn(Y) * dblPi / 2
    End If
  End If
End Function
```

Converting between Radians and Decimal Degrees

$$\text{Radians clockwise from North} = Degrees\left(\frac{\pi}{180}\right)$$

$$\text{Degrees clockwise from North} = Radians\left(\frac{180}{\pi}\right)$$

Spherical Radius Derived from Spheroid

When applying spherical functions, this extension assumes a sphere with the same volume as the actual data spheroid. For example, if the data used the WGS 84 spheroid (with semi-major axis = 6378137m, and semi-minor axis = 6356752.31424518m), then the spherical functions would be applied to a sphere with a radius of 6371000.79000915 meters derived as follows:

WGS 84 Spheroid:

Semi-major axis $(a) = 6378137.0$ m

Semi-minor axis $(b) = 6356752.31424518$ m

Volume of WGS 84 Spheroid:

$$V = \frac{4}{3}\pi a^2 b \quad (\text{WGS 84 is an oblate spheroid})$$

Volume of Sphere with Radius R:

$$V = \frac{4}{3}\pi R^3$$

Therefore solve for R when $\frac{4}{3}\pi R^3 = \frac{4}{3}\pi a^2 b$

$$R^3 = a^2 b \quad (\text{Like terms cancel})$$

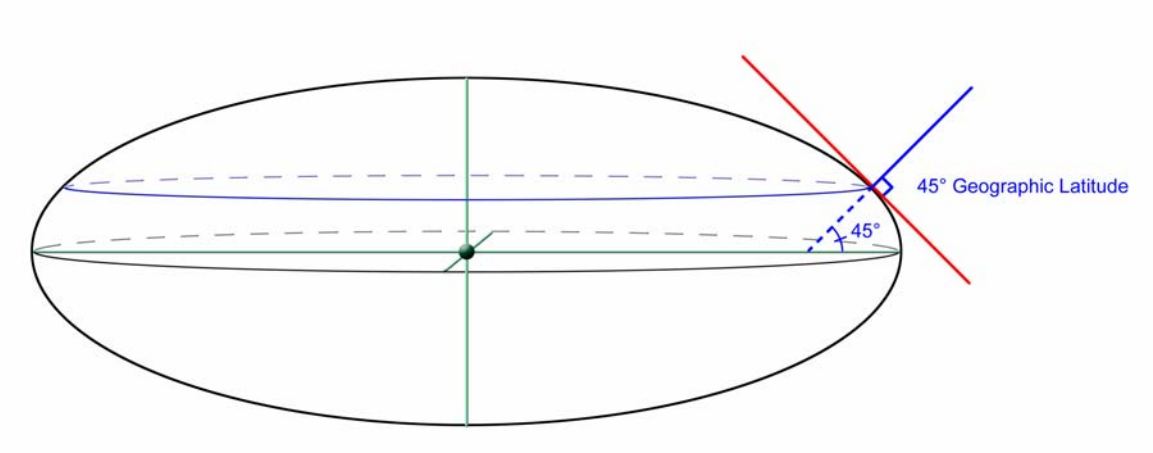
$$R = \sqrt[3]{a^2 b} = \sqrt[3]{\left(6378137.0^2\right)\left(6356752.31424518\right)}$$

$$= 6371000.79000915 \text{ m}$$

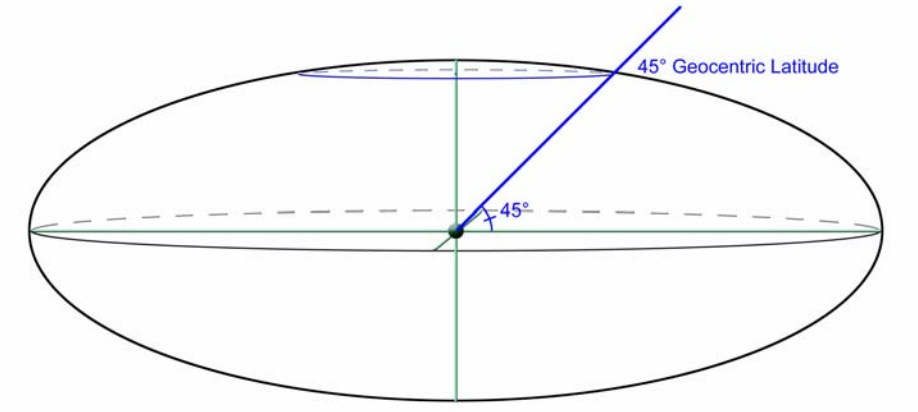
Planetocentric vs. Planetographic Coordinate Systems

For this tool, the terms Planetocentric (or Geocentric) and Planetographic (or Geographic, sometimes Geodetic) refer to the method of defining Latitude values on the surface of the spheroid.

Historically, latitude at a point on the earth has been determined by measuring the angle from the horizon at that point to some fixed reference point in space (the North Star, for example). Technically, this method produces the angle between the Normal of the spheroid (i.e. the vector perpendicular to the surface of the spheroid) at that point. Latitude coordinates produced by this method are referred to as Geographic (or Geodetic) coordinates. Planetary spheroids are typically slightly flattened spheres, so the Normal of the spheroid does not actually intersect the centroid of the planet. The illustration below greatly exaggerates the flattening to demonstrate this concept.



An alternative method would be to use the center of the spheroid as a reference point, and define "Latitude" as the angle between a line connecting the spheroid center to the point on the surface, and another line connecting the spheroid center to a point on the spheroid equator. Latitude coordinates produced by this method are referred to as Geocentric coordinates.



Because the prefix "Geo" implies Earth-based measures, I have substituted the prefix "Planeto" to create more general terms. **Note:** Most earth-based projections and datums are Planetographic, while most extra-terrestrial datums are Planetocentric.

Note: Longitude values are unaffected by whether the data are Planetocentric or Planetographic. The difference between Planetocentric and Planetographic latitude values depends on the flattening of the spheroid, where greater flattening leads to greater differences. If the spheroid

were a true sphere (with no flattening), then Planetocentric and Planetographic latitudes would be identical.

EQUATIONS

(Adapted from p. 17 of Snyder [1987])

$$\text{Planetocentric Latitude } (\psi) = \frac{180\left(\arctan\left(\left(\frac{b}{a}\right)^2 \tan\left(\frac{\phi\pi}{180}\right)\right)\right)}{\pi}$$

where ϕ = Planetographic Latitude

a = Semi-major axis radius

b = Semi-minor axis radius

$$\text{Planetographic Latitude } (\phi) = \frac{180\left(\arctan\left(\left(\frac{a}{b}\right)^2 \tan\left(\frac{\psi\pi}{180}\right)\right)\right)}{\pi}$$

where ψ = Planetocentric Latitude

a = Semi-major axis radius

b = Semi-minor axis radius

Revisions

Version 1.0

- Build 1.0.105 (December 30, 2008)
  - Extracted original “Calculate Geometry” function from “Tools from Graphics and Shapes”, under contract from the USGS to modify it for geocentric and non-terrestrial data.
  - Modified “Calculate Geometry” function to allow for ocentric/ographic transformations
  - Added tools to convert between ocentric and ographic data.
  - Added function to wrap longitude around pre-specified ranges (i.e. 0° to 360°, or -180° to 180°).

References

Chovitz, Bernard. 2002. In Memoriam: Thaddeus Vincenty. Available at http://www.gfy.ku.dk/~iag/newslett/news7606.htm. Last viewed December 28, 2007.

ESRI. 2006. ArcGIS 9 Media Kit: ESRI Data and Maps. County boundaries derived from US Census data. ESRI, Redlands, California, USA.

Iliffe, Jonathan. 2000. Datums and Map Projections for Remote Sensing, GIS and Surveying. Whittles Publishing, Scotland, UK. 150 pp.

Kennedy, Melita & Steve Kopp. 2000. Understanding Map Projections. ESRI, Redlands, California, USA. 110 pp.

Mardia, Kanti V., & Peter E. Jupp. 2000. Directional Statistics. John Wiley & Sons Ltd. West Sussux, England. 427 pp.

Snyder, John P. 1983. Map Projections Used by the U.S. Geological Survey, 2nd Ed. Geological Survey Bulletin 1532. U.S. Government Printing Office, Washington, DC, USA. 313 pp.

Snyder, John P. 1987. Map Projections – A Working Manual. Geological Survey Professional Paper 1395. U.S. Government Printing Office, Washington, DC, USA. 383 pp.

Veness, Chris. 2007a. Calculate distance, bearing and more between two latitude/longitude points. Available at http://www.movable-type.co.uk/scripts/latlong.html. Last viewed December 15, 2007.

Veness, Chris. 2007b. Vincenty formula for distance between two latitude/longitude points. Available at http://www.movable-type.co.uk/scripts/latlong-vincenty-direct.html. Last viewed December 15, 2007.

Vincenty, Thaddeus. 1975. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. Surv. Rev., XXII(176):88–93.

Wikipedia. 2007. Haversine formula. Available at: http://en.wikipedia.org/wiki/Haversine_formula. Last viewed December 25, 2007.

Williams, Ed. 2006. Aviation Formulary V1.43. Available at http://williams.best.vwh.net/avform.htm#Crs. Last viewed December 26, 2007.

Zwillinger, Daniel. 2003. CRC standard mathematical tables and formulae. CRC Press. www.crcpress.com. 912 pp.
