Efficient Ray Intersection with Global Terrain using Spheroidal Height-Augmented Quadtrees

Tech Report 98-38

Zachary Wartell, William Ribarsky and Larry Hodges

GVU Center, Georgia Institute of Technology

Revision 6/7/99 – Integrated several new figures based on reviewer comments on the shorter, published version of this paper:

Wartell, Zachary, William Ribarsky, Larry F. Hodges. "Efficient Ray Intersection for Visualization and Navigation of Global Terrain using Spheroidal Height-Augmented Quadtrees." VisSym '99, Joint EUROGRAPHICS - IEEE TCCG Symposium on Visualization, May 26-28, 1999, in Vienna, Austria.

Efficient Ray Intersection with Global Terrain using Spheroidal Height-Augmented Quadtrees

Zachary Wartell, William Ribarsky and Larry Hodges
GVU Center, Georgia Institute of Technology

Abstract

We present an algorithm for efficiently computing ray intersections with multi-resolution global terrain which is partitioned by spheroidal height-augmented quadtrees. While previous methods support terrain defined on a Cartesian coordinate system, our methods support terrain defined on a two-parameter ellipsoidal coordinate system. This curvilinear system is necessary for an accurate model of global terrain. Supporting multi-resolution terrain and quadtrees on this curvilinear coordinate system raises a surprising number of complications. We describe the complexities and present solutions. The final algorithm is suited for interactive terrain selection, simple collision detection and simple LOS (line-of-site) queries on global terrain.

1 Introduction

The increasing computation power, memory and rendering rates coupled with efficient data organization make it feasible to interactively visualize global 3D terrain with resolution down to a centimeter. Interactively rendering these large-scale terrain databases places increasing demands on the software system. Real-time level-of-detail management, efficient spatial subdivision and the use of a two-parameter ellipsoidal coordinate system (also called a geodetic coordinates) are a must.

This paper describes the impact of this geodetic coordinate system on quadtree spatial subdivision with respect to computing ray-terrain intersections. We extend a well-known ray-casting method for height-augmented quadtrees[1] defined on Cartesian coordinates. The extension handles multi-resolution terrain covered by height-augmented quadtrees which are based on geodetic coordinates.

2 Background

Our terrain visualization software is VGIS [Lind96]. VGIS uses automatic, continuous level-of-detail management for geometry and imagery and quadtree subdivision. To accurately model global terrain, VGIS uses a two-parameter ellipsoidal coordinate system commonly used in geodesy [Vani82].

This two-parameter ellipsoidal coordinate system is based on an oblate spheroid[2]. The two parameters are the spheroid's major semi-axis and minor semi-axis, *a* and *b*. *a* defines the X-Y dimension while *b* defines the Z dimension. In this system longitude, λ, is equivalent to the longitude in polar coordinates; however, latitude, Ψ, is the angle between the surface normal and the equatorial plane. Height, *h*, is measured parallel to the normal between the point in question and the underlying surface point (Figure 1).

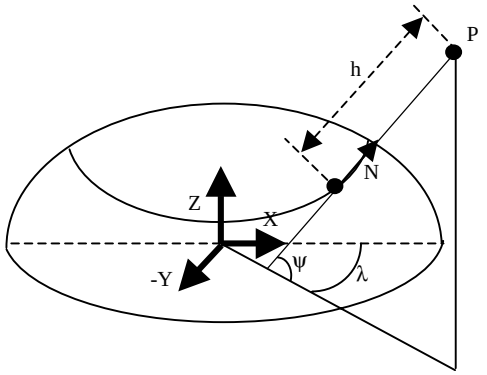


Figure 1: Spheroidal Coordinate System

VGIS builds its triangle mesh terrain database and its quadtree subdivision on this curvilinear coordinate system. The spheroid is first partitioned into 32 spheroidal quadrilaterals called zones. The zones are bounded by meridians and parallels. Each zone then contains its own quadtree. Each quadtree is further subdivided into quads which are also bounded by meridians and parallels. Overall, this divides the spheroid into triangles at the poles and quadrilaterals elsewhere. (Note that since parallels are not geodesics, these quadrilaterals and triangles are not true spheroidal quadrilaterals and spheroidal triangles; however, for brevity we will ignore this distinction). Finally each quad is augmented with a height attribute equal to the maximum spheroidal height of the contained data.

Using this terrain structure, we must provide an efficient method for finding arbitrary ray to terrain intersections. Such an algorithm serves as a basis for interactive terrain selection, collision detection and simple line-of-site queries.

While an efficient method for ray-casting through Cartesian coordinate height-augmented quadtrees is well-known [Coh93], this method assumes that the bounding volumes are bounded by their Cartesian

[1] A height-augmented quadtree simply adds a height attribute to each quad which is set to the maximum height value of the contained terrain.

[2] A spheroid is subclass of ellipsoid created by rotating an ellipse about its major or minor axis. It is synonymous with "rotation ellipsoid" [Drag82] ,"biaxial ellipsoid" [Vani82],"ellipsoid of revolution"[Smith97]

coordinate planes. Extending the algorithm to handle spheroidal height-quadtrees for multi-resolution terrain poses a number of problems. We present our solutions in order of their generality with respect how terrain is modeled. First we address tracing through the spheroidal bounding volumes. The presented algorithm applies to terrain modeled either as voxels, triangles, or bilinear patches. Next we address tracing through individual terrain elements. Here our solution is specific to terrain modeled as a regular triangle mesh. Third we address complications added by triangle mesh models which use multi-resolution data sets. Finally, we discuss surface continuity issues and discuss solutions that are specific to VGIS's continuity preservation methods.

3. Traversing Spheroidal Height-Quads

Cohen et al.'s [Coh93] method for efficient ray-terrain intersection, is similar in spirit to Bresenham line drawing. It traces the XY projection of a ray through the XY footprints of a height-augmented quadtree based on Cartesian coordinates. Upon entering a height-quad the entering and exiting z-coordinate of the ray is compared to the height of the quad. If the ray intersects the quad, the algorithm steps into the child quad at the next resolution level. Otherwise, the algorithm steps into the next quad at the same resolution level. The algorithm is so efficient that it is targeted towards real-time rendering of terrain. Figure 2 illustrates the high-level functionality of the algorithm. The figure is a side view with the ray in red and 3 levels of recursive height-quads. Blue volumes are intersected by the ray. Solid black volumes are not intersected, but the ray does enter their X-Y footprint. Dash black volumes are not examined by the algorithm at all. The red volume is the lowest level intersected volume. This figure illustrates the recursive nature of the bounding volumes and of the algorithm.

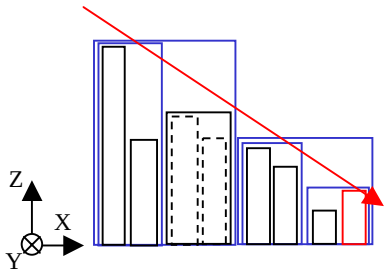


Figure 2: 2D illustration of Cartesian case.

Ideally, a spheroidal extension would use incremental integer calculations similar to Cohen's midpoint method. Unfortunately, while the basic high-level algorithm still applies, the midpoint technique that works so beautifully in the Cartesian setting appears to have no similarly efficient analog in the spheroidal case.

An exact analog would require a spheroid to plane mapping in which the spheroidal projection of a ray in 3-space maps to a line and in which the spheroid's quads are mapped to a regular square grid. The only common sphere to plane mapping that maps parallels, meridians and projected rays onto lines is the gnomonic projection [Mal73,p236]. The gnomonic mapping centrally projects the sphere through the sphere center onto a plane. The plane is placed tangent to the sphere at an arbitrary intersection point. Unfortunately, this mapping projects spherical quads onto planar rectangles with varying sizes. As we examine rectangles farther and farther away from the plane-to-sphere intersection point, the rectangles' areas grow towards infinity. The gnomonic map will not allow us to translate the Cartesian algorithm to the spheroidal case.

A partial analog to the Cartesian algorithm would require a spheroid to plane mapping in which quads map to a regular square grid and a projected ray maps to a curve. A cylindrical mapping can map quads onto a regular square grid. Unfortunately, a 3D ray projected onto the spheroid and then mapped to the plane by the cylindrical projection gives the following curve in latitude and longitude:

$$\lambda = \tan^{-1}\left(X \big/ Y\right) \qquad (3-1)$$

$$\psi = \tan^{-1}\left(Z + e'^{2}\, b \sin^{3}\theta \Big/ p - e^{2} a \cos^{3}\theta\right)$$

where

$$\theta = \tan^{-1}\left(\frac{aZ}{bp}\right)$$

$$p = \sqrt{X^{2} + Y^{2}}$$

$$e^{2} = \frac{a^{2} - b^{2}}{a^{2}}$$

$$e'^{2} = \frac{a^{2} - b^{2}}{b^{2}}$$

[*Hooijberg* 97, *page* 176]

and where X, Y, Z *are the ray coordinates in* 3 *space given by*

$$X = PX + VX\, t$$

$$Y = PY + VY\, t$$

$$Z = PY + VY\, t$$

While efficient methods for discretizing lines, ellipsis [Fol90], and cubics [Wat92] are known, a similarly efficient method of discretizing a curve of this complexity is not available.

For parallel-meridian height-quads, we must then resort to floating-point computation of ray-quad boundaries. Unfortunately, there appears to be no closed form solution for solving t in terms of ψ. This would be necessary for computing the projected ray's quad intersections with closed form arithmetic.

We therefore perform the ray-quad intersection tests in 3D dimensions where closed form solutions exist. We begin by describing the surfaces bounding a spheroidal height-quad. Generally these boundaries consist of 4 side boundaries formed by 2 plane wedges and 2 cone wedges and consist of upper and lower boundaries formed by quadrilaterals on the normal expansion of the spheroid (Figure 3).

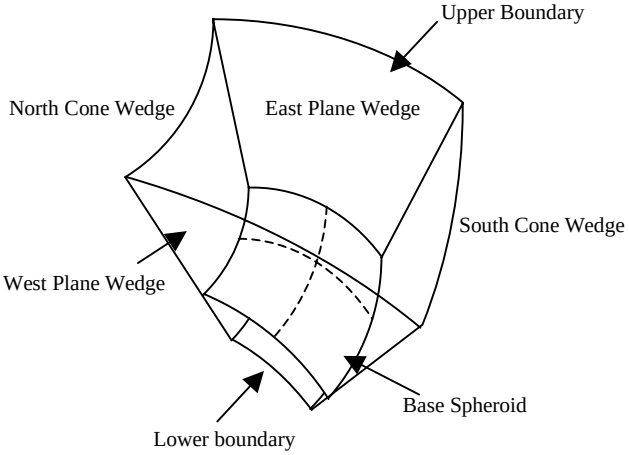


Figure 3: A spheroidal height-quad

3.1 Upper Bounding Surface

We show below that the upper bounding surface is not a spheroid nor an ellipsoid. Since a simple and efficient solution to computing ray intersections with this true upper boundary seems remote, we then derive an approximate spheroidal boundary covering this true boundary. This new boundary allows for simple computation.

3.1.1 TB^h: The True Bounding Surface

As previously mentioned, the spheroidal height-quadtree is based on an oblate spheroid S centered at a Cartesian coordinate system with spheroid's axes along the coordinate axes. The major axis of size a lies on X and Y axes while the minor axis of size b lies on Z axis. Each height-quad stores the maximum height h of the terrain it contains. This height is measured normal to the surface S. This maximum height defines a surface TB^h ('TrueBound') which bounds the height-quad. To describe TB^h begin by observing the X,Z plane. Let P be the points on the cross section, S_{xz}, of spheroid S, let T be the tangent and let N be the normal vector.

We can parameterize these points and vectors using a common ellipse parameterization on a parameter Θ. Intuitively, this parameterization is built by mapping our ellipse in our "real" space to a unit sphere in a scaled space where Θ is the polar coordinate in this scaled space. The scaled space to real space mapping is a scale by (a,b). Figure 4 illustrates this. (Note that Θ and θ are distinct).

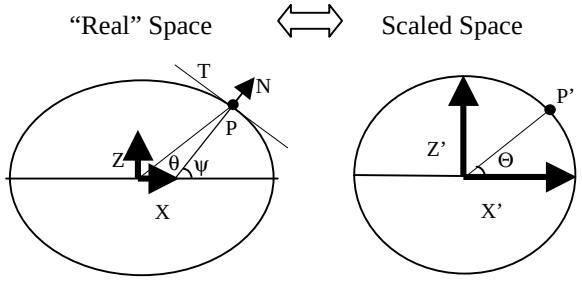


Figure 4: Parameterization of Ellipse

The parameterization yields:

$$P = (a\cos\Theta, b\sin\Theta) \qquad (3.1.1-1)$$

$$\Rightarrow T = (-a\sin\Theta, b\cos\Theta)$$

$$\Rightarrow N' = (b\cos\Theta, a\sin\Theta)$$

$$\| \; N' \cdot T = 0$$

$$\Rightarrow N = \left(\frac{b\cos\Theta}{D_\Theta}, \frac{a\sin\Theta}{D_\Theta}\right)$$

$$\text{where } D_\Theta = \sqrt{b^2\cos^2\Theta + a^2\sin^2\Theta}$$

$$\| \; N = N'/|N'|$$

Since a point $TB^h_{xz}(\Theta)$ on TB^h_{xz} is simply $P(\Theta)+hN(\Theta)$, TB^h_{xz} is described as follows:

$$TB^h_{xz}(\Theta) = \left(\left(a + \frac{hb}{D_\Theta}\right)\cos\Theta, \left(b + \frac{ha}{D_\Theta}\right)\sin\Theta\right) \qquad (3.1.1-2)$$

Unfortunately, TB^h_{xz} is not an ellipse. The unique ellipse passing through TB^h_{xz}'s extreme points (a+h,0) and (0,b+h) is $((a+h)\cos\Theta,(b+h)\sin\Theta)$. Since TB^h_{xz} does not equal this ellipse, TB^h_{xz} cannot be *any* ellipse. This further implies TB^h is not a spheroid nor even an ellipsoid.

Even worse, TB^h_{xz} contains degenerate cases for $h<0$ in which TB^h_{xz} can no longer function as a useful bounds for spheroidal height-quads. However, if we stipulate that $h>-b^2/a$ then these degenerate cases are removed (Appendix A2). In practice, for modeling surface terrain this stipulation is valid because $|h|<<a,b$ and b is close to a. For example using WGS_84 data for Earth, a = 6378137.0000, b = 6356752.3141 so $-b^2/a$ = -6335439.327003. The minimum geodetic terrain

height is several orders of magnitude smaller than this, around -15,000.

Ray-casting through height-quadtrees requires finding the intersection of a ray with the bounding surfaces of the height-quads. Unfortunately, an algebraic solution to the intersection of a ray and TB^h_{xz} seems remote even in two dimensions. Such a solution requires solving for a variable embedded in a complex expression involving the root of trig functions and square roots of trig functions. While an iterative approach might be used, that is complex and perhaps undesirably slow for our purposes.

3.1.2 B^h: An Approximate Bounding Surface

Since the height-quad is merely a *bounds* on the underlying terrain it is sufficient to use another surface B^h which bounds TB^h. B^h should allow an easy analytic solution to the intersection of a ray and B^h should 'closely' bound TB^h. An obvious choice for B^h is another spheroid.

Given the definition of TB^h and the fact that S is a sweep of the ellipse S_{xz} around the Z axis, it follows that TB^h equals TB^h_{xz} swept around the Z axis. Therefore B^h should be constructed with the same symmetry. With this in mind, we focus on B^h_{xz} and must find appropriate major and minor axes. Our solution is:

$$B^h_{xz}(\Theta) = \begin{cases} \left((a+h)\cos\Theta, \left(b + \frac{ha}{b}\right)\sin\Theta\right) & \text{if } h \geq 0 \quad (3.1.2-1) \\ \left(\left(a + \frac{hb}{a}\right)\cos\Theta, (b+h)\sin\Theta\right) & \text{if } h \in (-b^2/a, 0) \end{cases}$$

That this choice of B^h_{xz} bounds TB^h_{xz} can be proved rigorously (Appendix A3). Since B^h and TB^h are just rotational sweeps of B^h_{xz} and TB^h_{xz}, B^h must bound TB^h. Given the formula for B^h_{xz} computing B^h is straight forward.

3.2 Lower Bounding Surface

Since we can reasonably assume that traced rays begin outside the planet, it is sufficient to choose a single global lower bounding surface. The geodetic spheroid is inappropriate because it is chosen to best fit the planet's shape. Therefore terrain can exist both above and below the spheroid. We might use the global minimum geodetic height, but like the upper bounding surface, TB^h, the lower bounding surface defined by this height is not an ellipsoid. Moreover, we cannot use the aforementioned approximate bounding, B^h, surface because the approximate surface lies *outside* the true surface while an approximate lower boundary would require an approximation lying *inside* the true boundary. Therefore, we simply model the lower boundary as a sphere whose radius equals the distance from the spheroid center to the closest terrain vertex. Since typical traced rays start outside the terrain surface, choosing a global lower bound is acceptable even though it cannot bound individual height-quads as closely as separate, local lower bounds could.

3.3 Longitude Bounding Surface

On a spheroid the meridians are great elliptic arcs [Hooi97] and therefore by definition are embedded in a plane which intersects the spheroid's center. From the z-axis symmetry it is also intuitively obvious and easy to prove that the normals of the spheroid along a meridian are contained in the same plane. So terrain at a quad's longitude boundary is confined to this longitude's meridian plane. Specifically for a given longitude λ the associated plane is simply:

$$\cos\lambda \; x + \sin\lambda \; y = 0 \qquad (3.3-1)$$

The longitude boundary is a wedge from this plane.

3.4 Latitude Bounding Surface

Recall the definition of geodetic latitude ψ of a point P with surface normal N. Since ψ is defined by the angle between the line L defined by (P,N) and the line from the origin to L's intersection with the Z=0 plane (see Figure 1) and since the spheroid S is symmetric around the Z axis, it is easy to see that sweeping L about the Z axis yields a section of cone. The latitudinal bounding surface is a wedge of this cone, C.

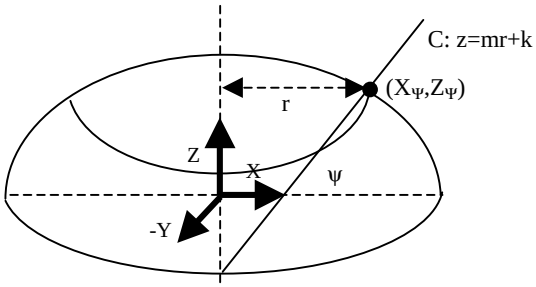


Figure 5: Derivation of latitude cone wedge.

Letting r be the cone radius for a given z and Z_ψ and X_ψ be the intersection of C with the y=0 cross section of the spheroid, yields a derivation of C (Figure 5):

$$z = mr + k \qquad (3.4-1)$$

where $m = \tan\Psi$

$$k = Z_\Psi - X_\Psi m$$

$$\Rightarrow z = m\sqrt{x^2 + y^2} + k$$

$$\Rightarrow C: x^2 + y^2 - \frac{z^2}{m^2} + \frac{2zk}{m^2} - \frac{k^2}{m^2} = 0$$

X_ψ and Z_ψ are calculated easily from [Drag82] (page 177) yielding:

$$X_\psi = \frac{a\cos\psi}{\sqrt{1-e^2\sin^2\psi}} \qquad (3.4-2)$$

$$Z_\psi = \frac{a\left(1-e^2\right)\sin\psi}{\sqrt{1-e^2\sin^2\psi}}$$

where

$$e^2 = \frac{a^2-b^2}{a^2}$$

3.5 The Algorithm

While the high-level principles of the Cohen algorithm (Section 3), apply to the spheroidal case, the details differ. The spheroidal algorithm is divided into two procedures. The user called procedure performs setup and zone traversal and calls a recursive procedure which recursively traverses through each zone's quadtree. The user called procedure first clips the ray to the volume bounded by a global upper boundary and the global lower boundary. The global upper boundary is the upper bounding surface, B^h, with height equal to the maximum global height. As part of this clipping, we compute, *t_global_exit*, the ray parameter value of the ray's global exit point. Next, we determine which zone contains the ray origin. Starting with this zone, we step through successive zones until either an intersection occurs or the ray exits the global boundaries. Zone traversal is quite similar to quad child traversal which is discussed in detail below. For each zone we then call the recursive quadtree procedure to traverse the zone's quadtree.

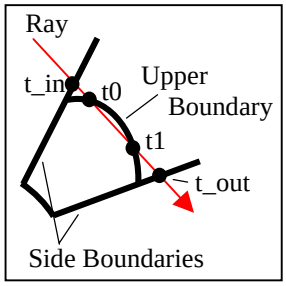


Figure 6: Illustration of upper boundary test.

The recursive procedure must first determine whether the ray, which is assumed to enter the current quad's side bounds, truly intersects the quad volume. Since the upper boundary is curved, it is insufficient to check the height of the ray's entering and exiting intersections with the side boundaries. Instead we compute the ray's parameter values, *t_in* and *t_out*, at these side intersections and we compute the ray's intersection parameters, *t_0* and *t_1*, with the quad's upper boundary surface (Figure 6). If and only if these two parameter intervals overlap, then the ray has entered the height-quad volume and we step through the quad's children.

If the quad volume is intersected, the algorithm must traverse the quad's children and recurse at each child. The first encountered child is determined from two factors. The first factor is which side boundary of the parent the ray entered. This factor is an argument, *side_in*, of the recursive quadtree procedure. The second factor is in which half-space of one of the parent's internal partition surfaces the entrance point lies. In Figure 7 these internal partitions are shown in blue. They partition the quad into four sections. The latitude partition surface is a cone wedge stretching east-west. The longitude partition is a plane wedge stretching north-south. Knowing which side boundary is entered and which internal partition half-space contains the entrance point, we know which child quad to visit. For example, in Figure 7 the ray (red) enters the west side boundary (i.e. *side_in* = WEST). So we test the entrance point (marked by the first red X) against the internal latitude partition. Since the entrance point is in the north half-space of this partition, the ray enters the north-west child. Four side boundaries and two partition half-spaces yield eight combinations. Each combination maps to one child. By determining which combination occurs, the algorithm determines which child to visit. Note an exception arises when the current quad contains the ray origin. In this case, *side_in* will have the value UNKNOWN. We must visit the child containing the ray origin. This child is determined by examining which half-spaces of both internal partitions contain the ray origin.

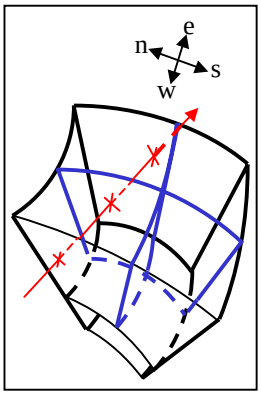


Figure 7: Illustration of child traversal

Having visited the first child, we must determine the other child quads intersected by the ray. Note that in the spheroidal case, a ray may intersect all four of a quad's children or may enter a quad twice. This can occur since a ray can have two intersections with a quad's latitude cone boundary. Given these

complications we determine the next child quad by computing the ray's intersection with the current child's boundaries. Note these boundaries are subsets of the parent's boundaries and the parent's internal partitions. The child exit boundary is the child boundary whose ray intersection's ray parameter value is the smallest while still being greater than the child's entrance point's value. This exit point is illustrated by the second red X in Figure 7. So given the current child, we compute *t_child_out*, the ray parameter where the ray exits the child, along with *side_out*, the boundary of the child at this exit. With knowledge of *side_out*, we know what child is entered next. For example, if the current child is the north-west child and *side_out* is found to be EAST, then the next child is the north-east child. Child traversal terminates when either a child reports a terrain intersection, all children are visited, or *t_child_in* of the current child is greater than *t_global_exit*, the ray intersection with the global upper boundary. Note as we step from one child to another we need not explicitly compute the next child's entrance ray parameter, *t_child_in*, nor its entrance side, *side_in*. *T_child_in* equals *t_child_out* of the previously encountered child while the entrance side, *side_in*, equals the complement of the previously encountered child's exit side, *side_out*.

Further efficiency can be gained during child traversal when determining *t_child_out* and *side_out*. We can limit the number of child boundaries whose intersections must be checked. For a quad in the northern hemisphere if a ray enters the west, east or north border, the side the ray next exits must differ from the entered side. We only need to test for exit intersections with 3 other sides. If a ray enters the south border, however, we must test all four sides for the exit point. For quads in the southern hemisphere the rule is reversed with respect to latitude borders; a south entrance requires only 3 intersection tests while a north entrance requires all 4.

4.0 Traversing Individual Terrain Elements

While the methods of the previous section apply to terrain regardless of the modeling method (voxel, bilinear patch, or triangles), the issues raised when traversing individual terrain elements are model dependent.

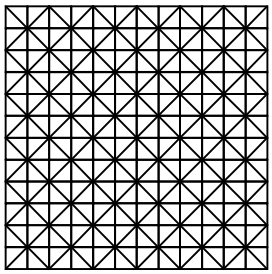


Figure 8: Matrix of Triangles

In voxel ray-casting methods [Coh93] the height-quad tree recurses down to the level of the smallest modeled terrain element. In regular triangle mesh methods, however, the height-quad tree typically does not recurse down to the level of the smallest modeled terrain element. Instead, a quad contains a fixed size matrix of triangles such as in Figure 8. Within this block there is no further quadtree subdivision. This means that for triangle modeled terrain, once we trace a ray to the bottom level quad, we must then separately trace the ray through that quad's block of triangles.

Additionally, the modeling method affects the mathematical surface in between the sampled elevation points. If we render with ray-casting we might model the surface as set columnar voxels which project radially out from the zero-elevation surface and are capped by either planar quads or spheroidal quads. (Note on the spheroidal coordinate system, these voxels are not cubes as in traditional Cartesian based terrain.) Alternatively, we can define the surface to be a set of bi-linearly interpolated patches. This is the typical method of interpolating height fields in geodesy [Hooi97,p198]. (Note, again, here we would bi-linearly interpolate spheroidal coordinates not Cartesian ones.) Unfortunately, while these are the most mathematically robust surface definitions, a practical polygon graphics system must interpolate between sampled elevation points by treating these points as triangle vertices.

Here we will focus on this triangle model. In order to minimize the number of triangles tested, we treat each triangle-pair as if it was contained in its own small height-quad and we then visit only those height-quads whose sides are intersected by the ray.

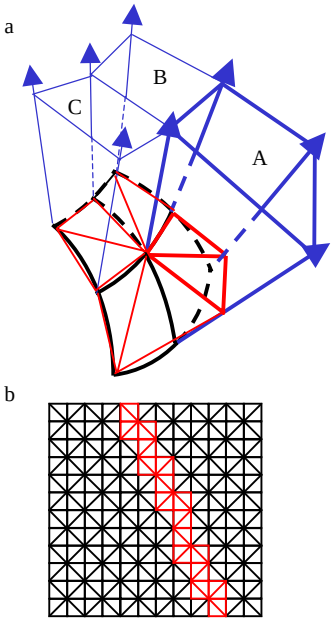


Figure 9: a) 4 triangle pairs (red) in a section of a quad (black) and the volumes which contain the 3 of the pairs (blue). The blue arrows are extensions of the spheroid normals. b) Using these volumes we can reduce the number of examined triangle pairs to those contained in intersected volumes. So instead of inspecting all triangle pairs in a quads mesh (black) we only inspect a subset (red).

In Figure 9a, four triangle pairs are drawn in red on a part of a spheroidal quad in black. The blue arrows are extensions of the spheroid normals at the quad's terrain grid points. Triangle vertices are confined to these lines. Furthermore, the blue lines delineate plane wedges defining four-sided volumes (blue). Note, triangle edges are confined to these plane wedges. These four-sided volumes can serve a similar purpose to the higher level height-quads. If the ray intersects the first triangle pair's volume, A, (in bold blue), we determine which of the 4 neighboring triangle-pairs to visit next by intersecting the ray with the volume's planar wedge sides. If the ray intersects the side shared by volume A and B, this tells us to visit the triangle-pair volume B. Similarly if the ray next intersects the side shared by B and C, we step into volume C. At each volume we test for ray-triangle intersections with the triangles in that volume. We continue this traversal until either a triangle is intersected, the quad boundary is reached or *t_volume_exit*, the ray parameter at its exit from the current triangle-pair's volume, is greater than *t_global_exit*. Figure 9b illustrates a typical pattern of examined triangle pairs in red.

Unfortunately the triangle model poses a theoretical problem that the other surface models do not have. Since the spheroidal height-quads are concave volumes, they will not contain all parts of the triangles whose vertices are contained in the quad volume and assigned to the quad. Specifically, the latitude conical boundaries do not contain all parts of the planar terrain triangles along the latitude border. This problem is illustrated in Figure 10.

Figure 10 shows 3 terrain triangles in red at the corner of a quad whose east, north, south and lower boundaries are drawn in black. The upper triangle is assigned to the illustrated quad while the lower 2 triangles are assigned to the adjacent quad across the south border. The green highlighted portion of the lower 2 triangles is the portion of these lower triangles not contained in the adjacent quad.

The containment problem can potentially cause the ray intersection algorithm to fail to discover an intersection. Referring to Figure 10, the ray could first pass over the adjacent southern quad without intersecting it and then enter the illustrated quad. If the ray is at a steep angle, it could then piece the green area. Since the illustrated quad does not contain the triangles associated with this green area, the ray will exit the global lower boundary and the algorithm would falsely indicate no intersection occurred.

Since this problem does not occur on the east-west planar meridian boundaries, one might suggest avoiding the parallel-meridian quadtree and using an alternative quadtree [Borg92][Feke90][Hwa93][Otoo93]. Such an alternative would avoid this boundary problem for an arbitrarily dense partitioning if and only if the bounding curves are planar and the plane embeds all the spheroid normals along these curves. Unfortunately, the only curves satisfying this requirement are meridians and the equator. This follows easily from Bowring's theorem concerning normal sections [Hooi97]. This theorem states that except for meridians and the equator there exist exactly two normal sections[3] between any points on the spheroid. This means there is no single, planar curve containing both end points and both normals. Clearly this is a prerequisite for a planar curve that contains both end points and *every* normal on the curve. Since meridians and the equator alone cannot generate a useful recursive partitioning, no existing spheroidal partitioning strategy can solve the containment problem.

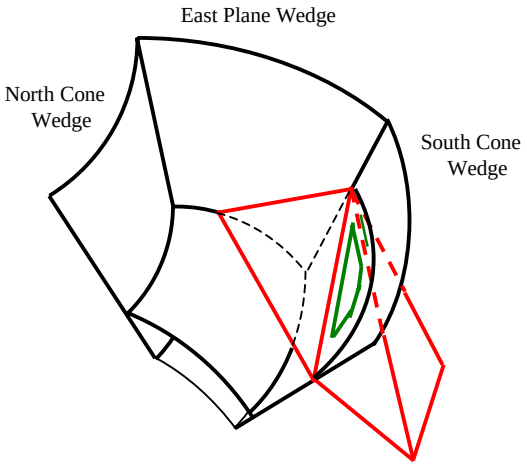


Figure 10: Shown here are 3 Sides and Lower Boundary of Spheroidal Quad (black) and 3 terrain triangles (red). The green portion of the lower two triangles is contained in the illustrated quad, but these triangles are assign to the adjacent quad to the south not to the illustrated one.

Another thought is to use rectangular bounding boxes instead of spheroidal height-quads. However, we must then deal with another set of complications. These boxes are laid on a spheroid surface and they bound terrain assigned to spheroidal quads. The bounding boxes of two quads which share a latitude border will overlap due to border curvature. Additionally, triangular quads at the spheroid poles that share a longitude border will also have overlapping bounding boxes. These boxes cannot cleanly partition space as do

[3] Recall, a normal section is a curve segment between two points on a surface that lies in a plane which contains the normal at one of the end points.

Cartesian bounding boxes on flat terrain [Coh93] or as do spheroidal height-quads on global terrain. This leads to ambiguities concerning the order in which child bounding boxes should be traced. Moreover, as discussed in 3.5 the proper order of spheroidal quad traversal is inherently more complex than the Cartesian case due to the curved boundaries, i.e. child quads can be visited twice or all four child quads can be visited. The overlapping bounding boxes on spheroidal terrain do not provide this ordering information.

Since these alternative partitioning methods do not help, we keep the parallel-meridian partitioning. Importantly, when using this partitioning and the outlined algorithm for interactively pointing at and grabbing terrain, it has been our experience that the pathological cases represented by the containment problem never occur [War99]. The reason is that each quad contains a relatively dense 128x128 triangle-pair block making the green area in Figure 10 extremely small. While the increasing curvature of the cone wedges at extreme latitude quads could exacerbate the containment problem, the increasing surface density of the triangles at these extreme latitudes counteracts this effect. This increase in surface density occurs because the quad surface area grows smaller at extreme latitudes.

5.0 Managing Multiresolution Aspects of Terrain

While covering the general traversal of the high-level spheroidal quads and the specific traversal of triangle-modeled terrain elements, we glossed over how a multi-resolution terrain model interacts with the ray casting algorithm. A typical multi-resolution model such as VGIS stores terrain data in 2^nx2^n blocks at resolutions at varying powers of 2. For rendering purposes, the system then goes to great lengths to ensure that the rendered terrain is a continuous surface. The algorithm uses a visual error metric to render the minimum detail level necessary to maintain visual quality while preserving mesh continuity [Lind96].

As previously mentioned, contrary to ray-casting models where the recursive subdivision of height-quads continues down to the level of individual voxels, in a triangle mesh model a leaf quad contains a N by N array of triangle pairs called a block. Equally important, however, the quadtree is not a full tree. Instead a branch is only as deep as necessary to reach the highest resolution block available in secondary memory. Moreover, while the complete quadtree is always in main memory, the actual triangle data at different quads are dynamically paged into main memory as dictated by the rendering algorithm. In Figure 11a, a flattened and zoomed in view of a single zone is shown. The outlines of the sub-quads existing in the zone are also shown. Higher resolution data is only available for the north-eastern most quads. This is indicated by the presence of the small quads in these regions. Figure 11b shows the corresponding quadtree data structure. The renderer pages-in triangle data associated with the smaller, higher resolution quads as it is needed in order to satisfy the visual quality constraints.

How the ray-intersection algorithm should deal with this dynamic paging is application dependent. For example if the user is viewing a battle scene with vehicles roaming over the terrain and he zooms out his view, VGIS may page out the high-resolution data for the battle space. If the ray-terrain intersection algorithm continues to be used for collision detection and line-of-site computation in the simulation, the algorithm could be forced to use the lower-resolution terrain now in primary memory. This could be very inappropriate as it could lead to arbitrarily large errors in vehicle collision detection and LOS computation. Clearly in this case, relying only on paged-in data is unsatisfactory.

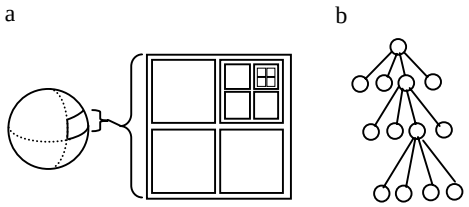


Figure 11: Illustration of an example quadtree from a single zone.

On the other hand, the ray-terrain intersection algorithm might be used to compute the intersection of a user held virtual laser pointer in a VR application. This would be a necessary step in interactive terrain positioning. In this instance, the ray-intersection algorithm must complete as quickly as possible. Therefore paging in new terrain data in response to ray traversal is a bad idea. The traversal should only access terrain data already in primary memory. (Of course, if even the lowest resolution data is unavailable for some traced region, we must perform paging.)

These complications lead to the following modification to the quad traversal algorithm. First we add a parameter controlling how ray traversal handles paged-out data. We add a parameter, *min_level* ("minimum level") that indicates what is the minimum quad node tree depth that may be used during triangle-pair traversal. Now we traverse the quadtree as detailed in 3.5 until either the ray exits the tree or the ray enters a leaf quad. If the ray enters a leaf quad, we need to find the highest resolution in-memory block that covers the quad and satisfies the *min_level* constraint. We find this block using a simple loop that steps through the leaf quad's ancestors. At each iteration we attempt to trace the ray through triangles of the ancestor quad. There are three possible results. Either the ray intersects a triangle, intersects no triangle, or the ancestor quad has

no in-memory data available. The loop continues until reaching an ancestor with data available or reaching an ancestor whose depth equals *min_level*. If we reach the *min_level* ancestor without finding paged-in terrain data, we must wait and page in this ancestor's terrain. Note that with this modification, the leaf quad depth could be than *min_level*. This means that the desired terrain data resolution simply does not exist in secondary storage. We must make do with the terrain data associated with this leaf by paging it in as needed.

Figure 12 illustrates an example. The diagonal red line is the ray. The quad nodes are color-coded. Gray nodes are not intersected at all. The black node's sides are intersected but the quad volume is not. A blue node's volume is intersected. A red node is a leaf node whose volume is intersected. A green node is a node whose triangle data is used for intersection testing. Figure 12 a and c show the tree structure. Solid circles indicate the node's triangle data is paged-in. Open circles indicate the data paged-out. For b let *min_level* be 0. The algorithm traces the ray down into the red node, but this node has no loaded data. The algorithm then traverses up to the solid green node which does contain data. This data is used for triangle traversal. For c let *min_level* be 1. The algorithm traces the ray down into the red node, but this node has no loaded data. The algorithm then traverses up the to open green node. Since *min_level* is 1 the algorithm must stop here. It cannot move up to level 0 and grab the solid blue node's data. Now triangle data must be loaded for the open green node.

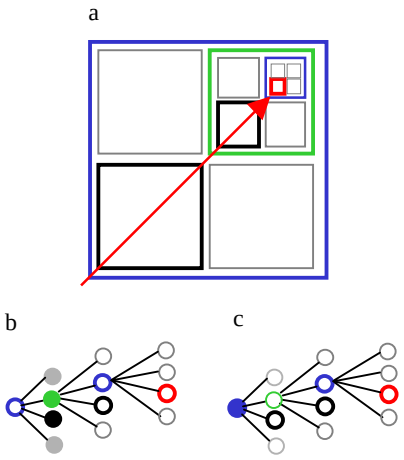


Figure 12: 2D illustration of quadtree traversal with triangle data loaded at different resolutions.

In general setting *min_level* to zero will use only the data in primary memory and will never wait for new data to be paged-in (unless even the lowest resolution data is absent). Setting *min_level* to the maximum possible quad tree depth will page-in whatever data is necessary to ensure that ray traversal uses the greatest resolution terrain in secondary storage. Setting *min_level* to some other value allows the programmer to trade off the resolution of the accessed terrain and the algorithm's performance.

A final detail is that when the appropriate ancestor is found we should only trace through the rectangular subset of its block which covers the original leaf quad. We compute the boundaries of this subset using an incremental integer approach. At the start of the ancestor loop the minimum indices of rectangular subset, *x_min* and *y_min*, are assigned zero. At each iteration of the loop, these indexes get either half their value or half their value plus half the block size. (Recall, block size is the number of vertices per row/column in a block and it is a constant value.) The choice depends on which child the current ancestor is relative its parent. If the current ancestor is a west child then $x\_min \leftarrow x\_min / 2$, otherwise $x\_min \leftarrow x\_min / 2$ + HALF_BLOCK_SIZE. If the current ancestor is a south child then $y\_min \leftarrow y\_min / 2$, otherwise $y\_min \leftarrow y\_min / 2$ + HALF_BLOCK_SIZE. The maximum corner of the rectangle is as follows: (x/y)_max = (x/y)_min + BLOCK_SIZE >> *levels*, where *levels* indicates the number of loop iterations. If 'BLOCK_SIZE >> *levels*' is zero, *x_max* and *y_max* are assigned one plus their respective minimum corner coordinates. We use this integer method since the floating point method allowed rounding errors that occasionally yielded invalid array indices. These boundaries are passed to the triangle tracing procedure which limits its traversal to the delimited subset of triangle-pairs.

6.0 VGIS Surface Continuity

Now we discuss complications due to surface continuity in the context of VGIS. When two adjacent terrain blocks have different resolutions the edges of triangles along the shared border will not match. When rendering, VGIS uses a set of rules to discard certain vertices and generate a triangle mesh using this vertex subset. This mesh has no cracks along block borders. How and/or when should we apply such rules to the terrain traversed by the ray intersection algorithm? Again the answer depends on the application of the intersection algorithm.

One option is to avoid such continuity rules and add artificial polygons between blocks to fill in the gaps. Whenever, the ray crosses block boundaries during triangle-pair traversal we compute temporary geometry for these polygons and test them for intersection with the ray. This method is the simplest but it violates the spirit of the VGIS rendering algorithm. However, it is fast and has worked well for terrain selection for VR navigation [War99].

If we apply continuity rules, we must used a modified version of VGIS's rendering rules. Instead of using the

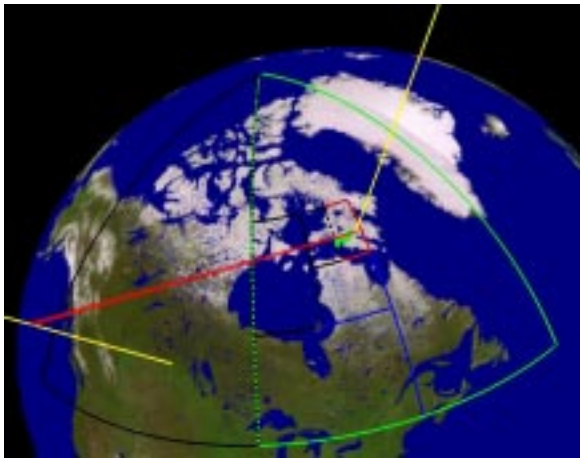
renderer's visual metric to determine vertex activation (i.e. its inclusion in the rendered mesh), we force all vertices to be active. We then apply the continuity rules to this fully activated mesh. This will generate the maximum resolution mesh that preserves continuity. This still leaves a big unanswered question. To what set of terrain blocks do we apply this algorithm? We can apply it dynamically to whatever blocks are paged in at ray-traversal time; or we can apply it off-line to the highest-resolution terrain and then force a continuity-requiring ray traversal to load in the highest resolution data. Next if we choose to apply the continuity algorithm dynamically we can apply the continuity algorithm to either the entire in-memory terrain dataset or only to the local regions accessed by ray traversal. Applying it to local regions, however, may produce a mesh differing from the mesh created by global continuity preservation. However, since VGIS is constantly paging data in and out, the mesh defined by applying dynamic global continuity is in continuous temporal flux. Therefore if we apply the continuity preservation dynamically, it may make little difference if we apply it locally or globally. Research to determine which of these options are appropriate to different continuity requiring applications is ongoing.

7.0 Results and Conclusions

Figure 13 illustrates the complete algorithm in operation. Here *min_level* is zero and we use the simplest continuity algorithm appropriate for fast interactive terrain selection. The application is running on a virtual workbench [War99] and the red ray is a virtual laser pointer interactively manipulated by the user. The two yellow lines indicate the projection of the ray origin onto the spheroid and the point on the ray where it exits the global boundaries. The visited height-quads' side boundaries are outlined in green, black, red and blue. Blue indicates the quad volume was intersected. Red indicates the quad was intersected and is a leaf. Green indicates the quad's polygon data was used for polygon traversal. Black indicates that while the quad side bounds were intersected the quad volume was not, i.e. the upper boundary was not pieced. The small streak of green inside the red quads are the outlines of the triangles that are tested for intersection. In Figure 13a, the planet is at a resolution such that the polygon data associated with the leaf quad (red) is not paged in. The algorithm visits ancestor quads until reaching the first quad (green) with polygon data covering the leaf quad. Figure 13b, shows a zoomed in view. Note here we can distinguish 2 sets of green polygons. The lower ones outline the actual tested triangles while the upper ones, are raised to the height of the source quad (green) and indicate the individual triangle-pair quads.

We are successfully using this algorithm for navigating global terrain on the virtual workbench [War99]. The algorithm is fundamental to the navigation method since the user navigates with a virtual laser pointer used to grab the terrain for panning, rotating and zooming. Empirically the intersection algorithm has had no affect on framerate as is necessary for VR interaction. Asymptotically, the algorithm is equivalent to standard quadtree and octree traveral methods.

a



b

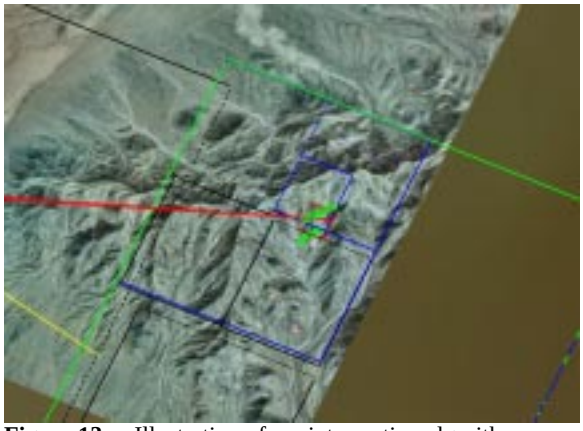


Figure 13: Illustration of ray intersection algorithm.

To conclude we have described the impact of the geodetic coordinate system on quadtree spatial subdivision with respect to computing ray-terrain intersections. We presented a new set of efficient methods for tracing a ray over the terrain. These methods go beyond the work of [Coh93], promoting a complete approach for global terrain in a multi-resolution spheroidal quadtree structure.

8.0 Future Work

There are several avenues of future work. First, while the containment problem has not been an issue for our current applications, it needs to be resolved for other applications. We are currently pursuing a hybrid

extension that mixes elements of spheroidal bounding volumes and Cartesian bounding volumes. The extension will remove the containment problem while keeping the number of visited quads to a minimum. Second, the continuity issues have yet to be fully resolved for all uses of ray-terrain intersection. Next it may be possible to switch from the spheroidal approach to the much simpler Cartesian approach [Coh93] when the algorithm reaches high detail quads. This is plausible because at some point the results of these two approaches will yield similar results. This is due to the finite precision of computer arithmetic. We are actively investigating these issues.

9.0 Acknowledgements

This work was performed in part under contracts N00014-97-1-0882 and N00014-97-1-0357 from the Office of Naval Research. Support was also provided under contract DAKF11-91-D-004-0034 from the U.S. Army Research Laboratory. We thank Frank Jiang for answering numerous questions about VGIS.

References

[Borg92] Gunilla Borgefors. A heirarchical 'square' tesselation of the sphere. Pattern Recognition Letters 13 (1992), pages 183-188.

[Coh93] Daniel Cohen-Or and Amit Shaked. Photo-Realistic Imaging of Digital Terrains. *Eurographics'93*, pages 363-373. 1993.

[Drag82] V.Dragonmir, , D.Ghitau, M. Mihailescu, M.Rotaru. *Theory of the Earth's Shape*. Elsevier Scientific Publishing Company. Amsterdam. 1982.

[Feke90] G. Fekete, L. Treinish. Sphere quadtrees: a new data structure to support the visualization of spherically distrsibuted data. Proceedings of the SPIE - The International Society for Optical Engineering. vol.1259. p.242-53. 1990.

[Fol90] James D. Foley, Andres Van Dam. *Fundamentals of Computer Graphics.* Addison-Wesley. Reading, Mass. 1990.

[Hooi97] Maarten Hooijerg. *Practical Geodesy Using Computers*. Springer. 1997.

[Hwa93] Sam C.Hwang, Hyun S. Yang. Efficient View Sphere Tessellation Method Based on Halfedge data Structure and Quadtree. Computer & Graphics, Vol. 17, No. 5 (1993), pages 575-581.

[Lind96] Peter Lindstrom, David Koller, William Ribarsky, Larry Hodges, Nick Faust, and Gregory Turner. Real-Time Continuous Level of Detail Rendering of Height Fields. *Computer Graphics (SIGGRAPH 96)*, pp. 109-118 (1996).

[Mal73] D.H. Maling, *Coordinate Systems and Map Projections*. George Philop and Son Limited. London. 1973.

[Otoo93] Ekow J.Otoo, Hogwen Zhu. Indexing of spherical surfaces using semi-quadcodes. Advances in Spatial Databases. Third International Symposium, SSD '93 Proceedings, pages.510-29.

[Smith97] James R. Smith. *Introduction to Geodesy*. John Wiley & Sons, Inc. 1997.

[Vani82] Petr Vanicek, Edward J. Krakiwsky *Geodesy, the concepts*. Amsterdam ; New York : North-Holland Pub. Co. ; New York, N.Y. : Sole distributors for the U.S.A. and Canada, Elsevier Science Pub. Co., 1982.

[War99] Zachary Wartell, William Ribarsky, Larry Hodges. Third-Person Navigation of Whole-Planet Terrain in a Head-tracked Stereoscopic Environment. To appear in *Proceeds of 1999 IEEE Virtual Reality Conference* (March 13-17 1999, Houston TX).

[Wat92] Ben Watson, Larry Hodges. Fast algorithms for rendering cubic surfaces. *Proceedings Graphics Interface '92* (May 11-15 1992, Vancouver, BC), 19-28.

Appendix:

In this appendix we tackle two issues. First we discuss degenerate cases of TB^h_{xz} and argue that in practice they do not occur. Second we prove that the upper boundary approximation, B^h_{xz}, truly bounds TB^h_{xz}. Both these proofs use several simple theorems that we first present in A1.

A1 Common Theorems

Theorem 1

For $\Theta \in [0,\pi/2]$ and a≥b and a,b>0, D_Θ monotonically increases over [b,a] and in general $D_\Theta \in$ [b,a].

Proof:

$$D_\Theta \in [b,a]$$
$$\Leftrightarrow \sqrt{b^2\cos^2\Theta + a^2\sin^2\Theta} \in [b,a]$$
$$\Leftrightarrow b^2\cos^2\Theta + a^2\sin^2\Theta \in [b^2, a^2] \qquad (1)$$
$$\| \text{since a, b} > 0$$

To prove this, examine the derivative of this expression to get local maximums and minimums:

$$\frac{\partial\, b^2\cos^2\Theta + a^2\sin^2\Theta}{\partial\Theta} = 0$$
$$-2b^2\sin\Theta\cos\Theta + 2a^2\cos\Theta\sin\Theta = 0$$
$$\left(2a^2 - 2b^2\right)\sin\Theta\cos\Theta = 0$$
$$\Rightarrow \Theta \in \left\{ n\pi/2 \;\middle|\; n \text{ is an integer} >= 0 \right\}$$
$$\| \text{ true since a} \geq \text{b}$$

Since the local maximum/minimum for interval [0,π/2] occur precisely at the interval endpoints it is sufficient to examine values at 0 and π/2. This yields simply b^2 and a^2 for the minimum and maximum proving (1) and hence $D_\Theta \in$ [b,a]. Finally, since no minimum or maximum occurs inside this interval we conclude D_Θ varies monotonically over [b,a] as Θ varies over [0,π/2].

Theorem 2

If a≥b and a,b>0, then $-a^2/b \leq -a \leq -b \leq -b^2/a$.

Theorem 3

If $h \geq -b^2/a$ and a≥b and a,b>0, then $a + hb/D_\Theta \geq 0$.

Proof:

$$1)\left(a + \frac{hb}{D_\Theta}\right) \geq 0$$
$$\Leftrightarrow h \geq -\frac{aD_\Theta}{b}$$
$$2) -\frac{ab}{b} \geq -\frac{aD_\Theta}{b} \geq -\frac{aa}{b} \qquad \| \text{Theorem 1}$$
$$3)\quad h \geq -\frac{b^2}{a} \Rightarrow \left(a + \frac{hb}{D_\Theta}\right) \geq 0 \qquad \| \text{1 \& 2 \& Theorem2}$$

Theorem 4

If $h \geq -b^2/a$ and a≥b and a,b>0, then $b + ha/D_\Theta \geq 0$.

Proof:

$$1)\left(b + \frac{ha}{D_\Theta}\right) \geq 0$$
$$\Leftrightarrow h \geq -\frac{bD_\Theta}{a}$$
$$2) -\frac{bb}{a} \geq -\frac{bD_\Theta}{a} \geq -\frac{ba}{a} \qquad \| \text{Theorem1}$$
$$3)\quad h \geq -\frac{b^2}{a} \Rightarrow \left(b + \frac{ha}{D_\Theta}\right) \geq 0 \qquad \| \text{1 \& 2}$$

A2 Degeneracies in TB^h_{xz}

We defined the true bounding surface as:

$$TB^h_{xz}(\Theta) = \left(\left(a + \frac{hb}{D_\Theta}\right)\cos\Theta, \left(b + \frac{ha}{D_\Theta}\right)\sin\Theta\right)$$

However, certain values of h<0 yield degenerate cases. To simplify discussion we restrict further discussion to a single quadrant, quadrant I, due to the symmetry of TB^h_{xz}. Degenerate examples for a base ellipse of a=10,b=11 are shown in Figure 14. In Figure 14a, the quadrant I of TB^h_{xz} is shown.. The largest curve is TB^0_{xz},, the base ellipse. Ordered diagonally from top-right to bottom-left are TB^h_{xz} for h=-100/11,-9.25,-9.75,-10,-10.5,-11,-11.5 and –12. Figure 14b, shows the complete surface for h=0 and then h=-100/11,-10.5,-12. Curve h=-100/11 appears reasonable, while curves h=-9.25,-9.75,-10,-11 exhibit a sharp corner in quadrant IV. Curves h=-11.5 and –12 exist in quadrants III and IV.

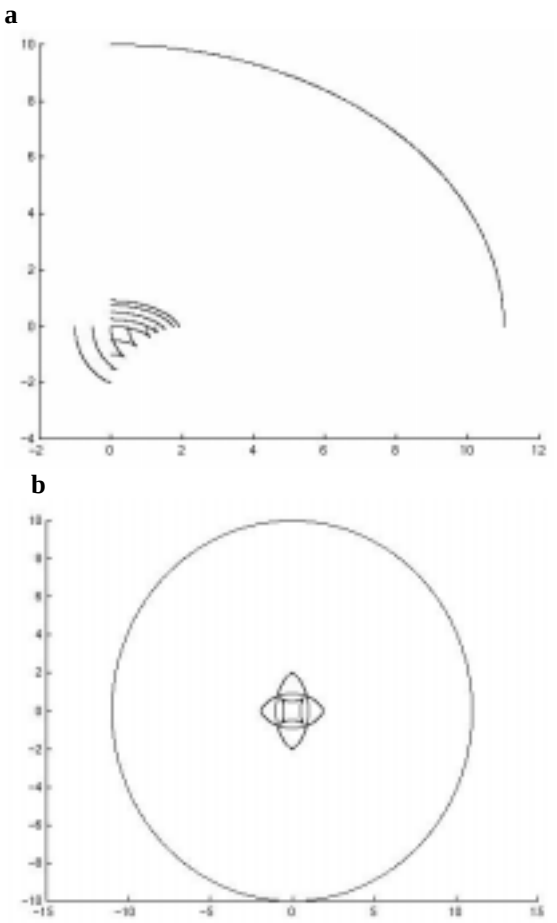
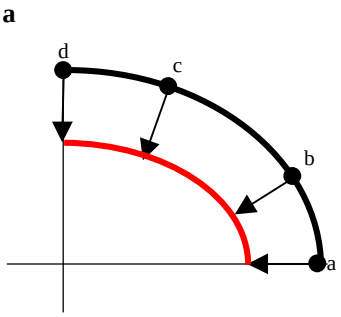


Figure 14: Degenerate Cases

We can better understand these degenerate cases by referring to the geometric construction of TB^h_{xz} in Figure 15. In Figure 15 quadrant I of the base ellipse is drawn (black) along with a corresponding section of TB^h_{xz} (red). At the labeled points, a,b,c and d, a vector is drawn. The vector is normal to the ellipse and of fixed magnitude corresponding to a negative height. We can imagine creating TB^h_{xz}, (red) by sweeping this vector along the base ellipse (black) and keeping the vector normal to the curve. 14a illustrates a non-degenerate cases while 14b and 14c illustrate degenerate cases.



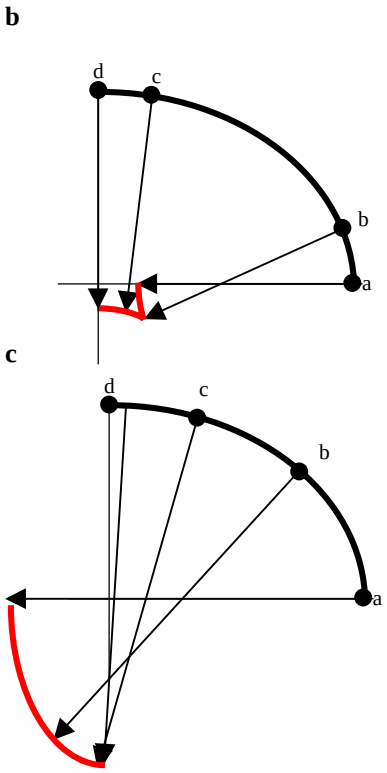


Figure 15: Construction of TB^h for non-degenerate cases (a) and degenerate cases (b) and (c).

The degenerate curves, in opposite quadrants and containing inflection points, make troublesome bounding surfaces. We like to avoid them by finding a lower bound for h above which they do not occur and by then showing that typical terrain data satisfies this bound.

The illustrated cases occur when the constructed point of TB^h_{xz} does not lie in quadrant I. It is easy to show that $h \geq -b^2/a$ implies the x and y coordinates of TB^h_{xz} are always positive or zero for $\Theta \in [0,\pi/2]$. The x coordinate is $(a+hb/D_\Theta)\cos\Theta$. Both factors are positive or zero on $\Theta \in [0,\pi/2]$ from Theorem 3. The y coordinate is $(b+ha/D_\Theta)\sin\Theta$. Both factors are positive or zero on $\Theta \in [0,\pi/2]$ from Theorem 4.

Having shown $h \geq -b^2/a$ avoids the degenerate cases, we now argue that for typical terrain surface models $h \geq -b^2/a$ is indeed true. This occurs since typically $h << a,b$ and a and b are close. For example, in the WGS_84 model of Earth, a= 6378137.0000, b= 6356752.3141 so $-b^2/a$ = -6,335,439.327003. The minimum geodetic terrain height is several orders of magnitude smaller than this, around -15,000.

Finally, due to a technicality when choosing an elliptical approximation to TB^h_{xz}, we make a minor modification and stipulate $h > -b^2/a$.

A3 B^h_{xz} Bounds TB^h_{xz}

Here we prove that our choice of B^h_{xz}:

$$B^h_{xz}(\Theta) = \begin{cases} \mathrm{B}_{h \geq 0}, \text{ when } h \geq 0 \\ \mathrm{B}_{h < 0}, \text{ when } h \in (-b^2/a, 0) \end{cases}$$

where

$$\mathrm{B}_{h\geq 0} = \left((a+h)\cos\Theta, \left(b + \frac{ha}{b}\right)\sin\Theta \right)$$

$$\mathrm{B}_{h<0} = \left(\left(a + \frac{hb}{a}\right)\cos\Theta, (b+h)\sin\Theta \right)$$

bounds TB^h_{xz}:

$$TB^h_{xz}(\Theta) = \left(\left(a + \frac{h\,b}{D_\Theta}\right)\cos\Theta, \left(b + \frac{h\,a}{D_\Theta}\right)\sin\Theta \right)$$

where $D_\Theta = \sqrt{b^2\cos^2\Theta + a^2\sin^2\Theta}$

given our standard assumptions that a≥b and a,b>0.

Theorem 5

If h≥0, then curve TB^h_{xz} lies inside or on $B_{h\geq 0}$.

Proof:

We can use the implicit equation of an ellipse $B_{h\geq 0}$ and prove that all points on TB^h_{xz} are inside or on $B_{h\geq 0}$. This is true if and only if:

$$\frac{X^2}{(a+h)^2} + \frac{Y^2}{\left(b + \frac{ha}{b}\right)^2} - 1 \leq 0$$

where

$$X = TB^h_{xz}(\Theta)[X], Y = TB^h_{xz}(\Theta)[Y]$$

$$\Leftrightarrow P(\Theta)^2\cos^2\Theta + Q(\Theta)^2\sin^2\Theta \leq 1$$

where

$$P(\Theta) = \frac{\left(a + \frac{h\,b}{D_\Theta}\right)}{(a+h)}, Q(\Theta) = \frac{\left(b + \frac{h\,a}{D_\Theta}\right)}{\left(b + \frac{ha}{b}\right)}$$

Now since $\cos^2\Theta + \sin^2\Theta = 1$ and both of these terms are positive, and since both $P(\Theta)$ and $Q(\Theta)$ are positive, it is sufficient to show that $P(\Theta)$ and $Q(\Theta)$ are both less than or equal to 1 for $\Theta \in [0, 2\pi]$.

That $P(\Theta) \leq 1$ is as follows:

$$P(\Theta) \leq 1$$

$$\Leftrightarrow \left(a + \frac{h\,b}{D_\Theta}\right) \leq (a+h)$$

$$\Leftrightarrow b / D_\Theta \leq 1 \quad \| \text{ true since a, b, h} > 0 \qquad (4-1)$$

$$\Leftrightarrow D_\Theta \geq b \quad \| \text{ true from Theorem 1}$$

Next prove that $Q(\Theta) \leq 1$:

$$Q(\Theta) \leq 1$$

$$\Leftrightarrow \left(b + \frac{h\,a}{D_\Theta}\right) \leq \left(b + \frac{ha}{b}\right)$$

$$\Leftrightarrow b / D_\Theta \leq 1 \quad \| \text{since a, b} > 0, \text{h} \geq 0$$

$$\Leftrightarrow D_\Theta \geq b \quad \| \text{ true from Theorem 1}$$

Theorem 6

If h∈(-b^2/a,0), then curve TB^h_{xz} lies inside or on $B_{h<0}$.

Proof:

Again, we use the implicit equation of an ellipse $B_{h<0}$ and prove that all points on TB^h_{xz} are inside or on $B_{h<0}$. Similar to Theorem 5, this is true if and only if:

$$\Leftrightarrow P(\Theta)^2\cos^2\Theta + Q(\Theta)^2\sin^2\Theta \leq 1$$

where

$$P(\Theta) = \frac{\left(a + \frac{h\,b}{D_\Theta}\right)}{\left(a + \frac{hb}{a}\right)}, Q(\Theta) = \frac{\left(b + \frac{h\,a}{D_\Theta}\right)}{(b+h)}$$

Now since $\cos^2\Theta + \sin^2\Theta = 1$ and both of these terms are positive, it is sufficient to show that $P(\Theta)$ and $Q(\Theta)$ are both less than or equal to 1, assuming both $P(\Theta)$ and $Q(\Theta)$ are positive or zero. Now $P(\Theta) \geq 0$ for h>-b^2/a since the numerator is ≥ 0 (Theorem 3) and the denominator is > 0 (Theorems 2 which show h>-b^2/a implies h>–a^2/b). $Q(\Theta) \geq 0$ for h>-b^2/a since the numerator is ≥ 0 (Theorem 4) and the denominator is > 0 (Theorems 2 which show h>-b^2/a implies h>-b).

So continuing to show $P(\Theta)$ and $Q(\Theta)$ are both less than or equal to 1, we begin with $P(\Theta) \leq 1$:

$P(\Theta) \leq 1$

$$\Leftrightarrow \left(a + \frac{h\,b}{D_\Theta} \right) \leq \left(a + \frac{hb}{a} \right)$$

$$\Leftrightarrow D_\Theta / a \leq 1 \quad \| \text{since } h < 0$$

$$\Leftrightarrow D_\Theta \leq a \quad \| \text{ true from Theorem 1}$$

Next prove that Q $(\Theta) \leq 1$:

$$Q(\Theta) \leq 1$$

$$\Leftrightarrow \left(b + \frac{h\,a}{D_\Theta} \right) \leq (b + h)$$

$$\Leftrightarrow a / D_\Theta \geq 1 \quad \| \text{since h} < 0$$

$$\Leftrightarrow D_\Theta \leq a \quad \| \text{ true from Theorem 1}$$

We conclude, that if $h \in (-b^2/a,0)$, then curve TB^h_{xz} lies inside or on the $B_{h<0}$.