

# Byning the Earth

Robert.G.Chamberlain @jpl.nasa.gov, (818) 306-6484

Jay.E.Braun@jpl.nasa.gov, (818) 306-6280

Paul .J.Firnett@jpl.nasa.gov, (81 8) 306-6485

Jet Propulsion Laboratory  
4800 Oak Grove Drive  
Pasadena, CA 91109

FAX: (81 8) 306-6953

Keywords: military applications, terrain representation, byning, battlefield simulation,  
Corps Battle Simulation (CBS)

The work described herein was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the U.S. Army Simulation, Training, and Instrumentation Command through an agreement with the National Aeronautics and Space Administration.

**THIS DRAFT IS FOR REVIEW PURPOSES ONLY. THE CONTENT IS NOT CLASSIFIED.**

**NONE OF THE ABOVE ORGANIZATIONS HAVE AUTHORIZED PUBLIC RELEASE.**

## Abstract

The Corps Battle Simulation (CBS) simulates ground and air battles between modern military forces. It runs in real time, modeling the interactions of tens of thousands of units in support of U.S. Army training exercises, typically involving several sites at geographically separated locations around the world.

Until recently, the CBS terrain database has associated terrain characteristics with a regular grid of three-kilometer hexagons tiling an appropriate map projection of the playbox. The hexagons also served as bins for other location-dependent information to simplify the search for *what's nearby*.

This paper presents a new way to describe locations: *Binary latitudes* and *longitudes* define patches of various sizes that completely cover the globe. Larger patches, not necessarily all the same size, can serve as bins. Medium-sized patches can be assumed to have uniform terrain characteristics throughout. Tiny patches are suitable descriptors of locations.

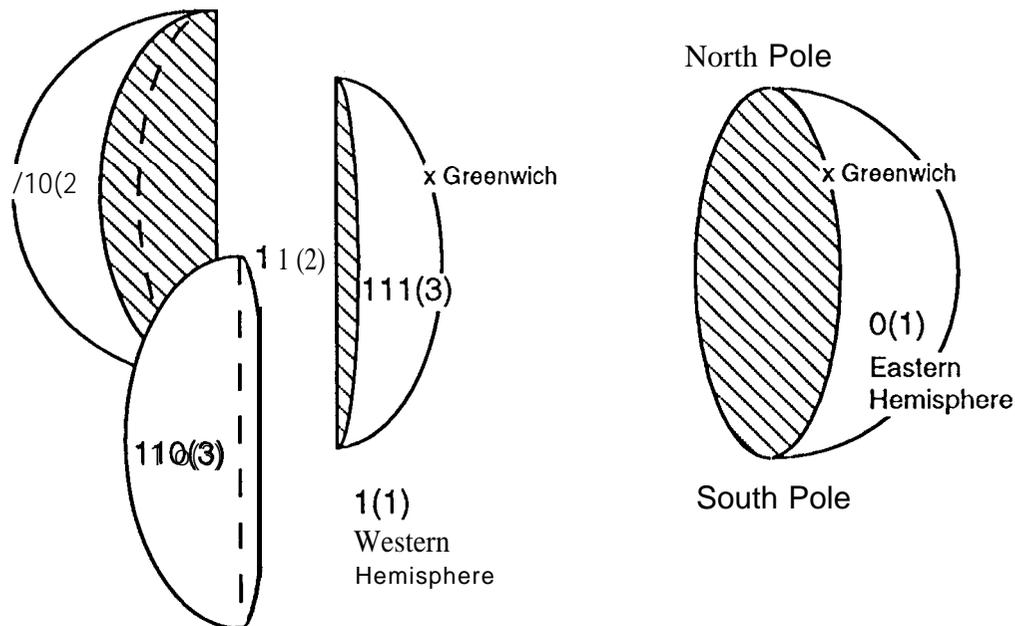
The new paradigm maybe implemented in CBS in the next development cycle. Spherical coordinates (latitudes and longitudes) are used for fundamental specification of locations, so playboxes are not limited by the characteristics of any map projection. Computation speed is not affected adversely, as flat-earth approximations are at least as valid as before. The natural data structure is a tree, which facilitates storing terrain (and other location-dependent) data in bins of variable size. If desired, huge bins can be used for areas remote from the primary theater of operations and relatively small bins for battle zones. Terrain patches need not have the same size or shape as the bins in which they are located.

## Summary

We have devised a new way to define locations and to describe patches of terrain: Successive bisections of the earth generate *byns*, coordinates that specify an area (rather than a point), establishing *binary* representations of latitudes and longitudes. Byns with high *fineness* numbers imply small areas.<sup>1</sup>

Byns can also define bins to be used to facilitate efficient searches for *what's nearby*. Bins may contain several terrain patches.

Byns are defined in a manner that is very natural for computers. Hence, it should be possible to develop low-level utility routines that are very efficient and compact. For example, finding the bin in which a location lies can use an efficient tree search algorithm. Creation and parsing of binary latitudes and longitudes is all but native to digital computers, should be extremely fast, and can be completely transparent to users. However, although binary representation of byns is very useful to understanding the concept and is expected to facilitate implementation, it is not requisite for validity or usefulness.



### Paradigm: Binary coordinates define an area, not a point.

Consider successive bisections of the range of possible longitudes, as illustrated in the figure above. Keep track of whether the first or second piece was used by a string of zeroes and ones—a binary number. Also keep track of how many bisections were required—the length of the string of zeroes and ones. If we store the string in the left side (up to 24 bits) of a 32-bit computer word, the *fineness* (the length of the string) can be stored in the final 8 bits of that word. Call this word

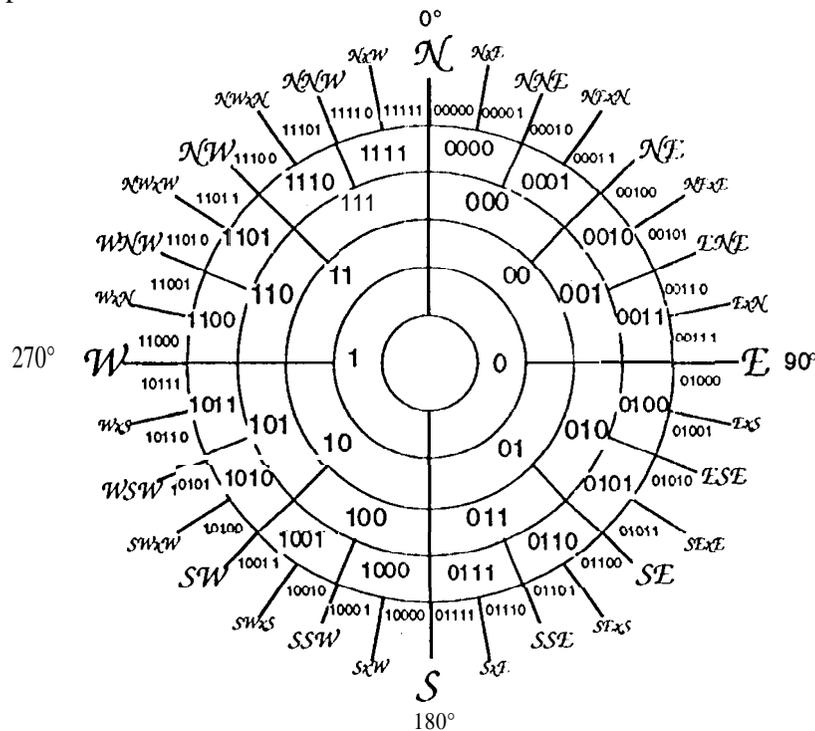
<sup>1</sup>*Byn* is an invented word. (The y is always pronounced long.) Its vague resemblance to the words bin, *byte*, and *binary* is, as will become apparent, intentional. A *binary* latitude is a band bounded by parallels of latitude, while a *binary* longitude is a zone bounded by meridians of longitude. The angular width of the band or zone is dictated by the byn's *fineness*.

a *bynary longitude*. The westward edge of this zone is obtained by interpreting the left side of the bynary longitude, the angular width by interpreting the fineness.<sup>2</sup>

The first bisection creates a piece, O(1), that extends from 0°E to 180°E (the eastern hemisphere) and a piece, 1(1), from 180°W to 0°W (the western hemisphere). Bisecting the 1 (1) piece creates pieces 10(2), from 180°W to 90°W, and, 11(2), from 90°W to 0°W. Bisecting the 11(2) piece creates pieces 110(3), from 90°W to 45°W, and 111(3), from 45°W to 0°W. And so on.

Create a *bynary latitude* by similar bisections of the range of possible latitudes, with the southern hemisphere being the first piece and the northern hemisphere being the second. However, latitudes span only 180° of arc, while longitudes span 360°. For convenience, we'd like each position in a byn to translate to the same angular width, so the first bit of a bynary latitude will always be O.

Bynary latitudes start at the South Pole, increase northward, and end at the North Pole. Bynary longitudes start at the Prime Meridian (the one through Greenwich Observatory), increase eastward, and complete the circle back at the Prime Meridian. Neither the text labels °N, °S, °E, or °W nor a mathematical sign, needed in conventional expressions of latitude and longitude, are required. The procedure is similar to *boxing the compass*, as illustrated in the figure below. In both cases, the complete circle is successively bisected, but a byn defines a portion of the circle, while a compass point defines a direction.



<sup>2</sup>The resultant number can conveniently be viewed in a hexadecimal (rather than binary) format, with the resolution (the final 8 bits, the *fineness*) enclosed in parentheses. For example, the longitude byn 000000(01) refers to the entire eastern hemisphere, while 000000(18) refers to a zone that is 1/24th as wide as one-tenth of a second of arc wide and passes through Greenwich Observatory. (Recall that 24 in decimal notation is represented as 18 in hexadecimal.)

### Parsing: How to create or interpret a byn.

A byn is created by successive bisections of a circle. Thus, each position has a significance very similar to that of each digit in a binary number. In fact, a byn is a binary fraction of a circle, with an implicit binary (not *decimal*) point at the left.<sup>3</sup>

p, fineness or binary position	radians $2\pi/2^p$	degrees minutes seconds	meters (but only on a great circle)
1	3.141592654	180° 0' 0'	20,014,427
2	1.570796327	90° 0' 0"	10,007,214
3	0.785398163	45° 0' 0"	5,003,607
4	0.392699081	22° 30' 0"	2,501,803
5	0.196349540	11° 15' 0"	1,250,902
6	0.098174770	5° 37' 30'	625,451
7	0.049087385	2° 48' 45"	312,725
8	0.024543692	1° 24' 22.5"	156,363
9	0.012271846	0° 42' 11.25"	78,181
10	0.006135923	0° 21' 5.625"	39,091
11	0.003067961	0° 10' 32.8125"	19,545
12	0.001533980	0° 5' 16.40625"	9,773
13	0.000766990	0° 2' 38.20313"	4,886
14	0.000383495	0° 1' 19.10156"	2,443
15	0.000191747	0° 0' 39.55078"	1,222
16	0.000095873	0° 0' 19.77539"	610.8
17	0.000047936	0° 0' 9.88770"	305.4
18	0.000023968	0° 0' 4.94385"	152.7
19	0.000011984	0° 0' 2.47192"	76.35
20	0.000005992	0° 0' 1.23596"	38.17
21	0.000002996	0° 0' 0.61798"	19.09
22	0.000001498	0° 0' 0.30899"	9.544
23	0.000000749	0° 0' 0.15450"	4.772
24	0.000000374	0° 0' 0.07725"	2.386

Fineness

or

Contribution To Distance North Of South Pole (Bynary Latitude)  
Or East Of The Prime Meridian (Bynary Longitude)

To create bynary latitude and longitudes from conventionally described latitudes and longitudes: First, determine the desired finenesses. Next, since conventional latitudes are

<sup>3</sup>The great circle distance between two locations is simply the product of the radius of the earth, 6371 kilometers, and the subtended central angle expressed in radians. Of course, since the earth is not quite a sphere, any statement of its "radius" to this many significant figures is true only at a few places on the earth's surface. Give or take 20 kilometers or so, however, this number is correct everywhere. When the great circle arc is expressed in byns, the distance is the product of the byn and the circumference of the earth, 40 029\* 125 kilometers; conversion to radians is not necessary.

expressed as positive numbers of degrees north or south of the equator, while binary latitudes start at the South Pole, increase the latitude by  $90^\circ$  in the northern hemisphere, or subtract the latitude from  $90^\circ$  in the southern hemisphere. Further, since conventional longitudes are expressed as positive numbers of degrees east or west of the Prime Meridian, while binary longitudes are all east of the Prime Meridian, longitudes in the western hemisphere must be subtracted from  $360^\circ$ . Computationally, the remaining two steps are trivial: Divide by  $360^\circ$  to get a fraction of a circle, already in binary notation if the computation is performed on most digital computers, then discard the binary point.<sup>4</sup>

The process performed by the computer is long division, using binary arithmetic, which can be described in this context as follows (an example follows in the next paragraph): Initialize the remainder to the latitude (or longitude) after modification as described above. The first trial divisor is  $360^\circ/2$ , or  $180^\circ$ . If the remainder does not exceed the trial divisor, the bit is 0. If it does exceed the trial divisor, the bit is 1, and the trial divisor is subtracted from the remainder. Until the number of bits you have accumulated equals the fineness, repeat the process, with each trial divisor half the previous one.

latitude binary digit	remainder	degrees associated with binary digit	longitude binary digit	remainder
	142° 31'			13° 24'
0		180°	0	
1	52° 31'	90°	0	
1	7° 31'	45°	0	
0		22° 30'	0	
0		11° 15'	1	2° 9'
1	1° 53.5'	5° 37.5'	0	
0		2° 48.75'	0	
1	0° 29.125'	1° 24.375'	1	V 44.625'
0		0° 42.1875'	1	0° 2.4375'
1	0° 8.03125'	0° 21.09375'	0	
0		0° 10.546875'		
		0° 5.2734375'		

Consider, for example, the binary expression of the latitude and longitude of Berlin, located at  $52^\circ 31'$  N latitude,  $13^\circ 24'$  E longitude. A latitude fineness of 11 bits corresponds to  $10.5'$  of arc, a bit under 20 kilometers in north-south extent. Since a degree of longitude is only 60% as long at Berlin's latitude, fewer bits are needed in longitude: 10 bits produce an east-west extent a little under 24 kilometers. This is a reasonable precision for a city the size of Berlin. The following table illustrates the calculation. When the number in the "degrees associated with binary digit"

<sup>4</sup>In decimal notation, the result of dividing  $90^\circ$  by  $360^\circ$  is represented as .25, which is two times  $1/10^1$  plus five times  $1/10^2$ . (The optional zero to the left of the decimal point is not a "significant" digit and has been omitted.) The quotient is represented as .01 in binary notation. That is, it is zero times  $1/2^1$  plus one times  $1/2^2$ . Discarding the binary point gives the swing 01 to represent the angle.

column is larger than the current "remainder", make the bynary digit 0 and go to the next row. Otherwise, make the digit 1 and reduce the remainder accordingly. The resultant bynary latitude for Berlin is 01 100101010(101 1) or, in the (perhaps) more readable 32-bit hexadecimal form, 654000(0B), with 0s inserted in the undefined 13 bits. The bynary longitude is 098000(0A).

Note that the remainders were truncated, rather than used for rounding. This is a consequence of the fact that the bynary latitudes and longitudes define bands and zones, not points: the number 654000(0B) says that Berlin lies between 52°22'58.125" N latitude (the southern edge of the band) and 52°33'30.9375" N latitude (the southern edge of the band plus its width) and 098000(0A) says Berlin is in the zone bounded by 13°21'33.75" E longitude (the western edge of the zone) and 13°42'39.375" E longitude (the western edge of the zone plus its width). The selected finenesses can be interpreted as expressing the uncertainty in our knowledge of the location of Berlin, as the resolution of our statement of that location, or as our statement of the size of Berlin.

To interpret a bynary latitude or longitude, multiply the byn by 360° to get the latitude angle northward of the South Pole or the longitude angle eastward of the Prime Meridian. This multiplication is equivalent to summing the numbers from either of the above tables that correspond to each position in the byn that contains a one, up to the fineness of the byn. The result is the parallel of latitude or the meridian of longitude that corresponds to the southern or western edge of the patch. The angular width of the patch is given by the row of the table corresponding to the fineness. The same procedure can be used to interpret the central angle of any great circle arc; multiplication of the byn by the circumference of the earth, expressed in any units, gives the length of the arc in the same units.

### Bins and patches.

In all versions of CBS to date, three-kilometer hexes that tile the playbox have been used as "bins", owning lists of various simulation objects that occupy or arc associated with the hex. These lists have permitted efficient searches for "nearby" objects. Until CBS 1.4, the hexes (or their triangular sectors) were the lowest level of resolution played, so it was convenient to assign terrain characteristics to them. As fidelity in CBS has improved, resolution has increased. Bins that arc sized for efficient searching are too large to be considered to have uniform terrain. Thus, the bins will contain lists of terrain patches as well as of other simulation objects. Bins and terrain patches can be defined by the same process, carried to different levels of fineness.

Searches for nearby objects arc likely to encompass the current bin and those adjacent to it, which can be determined by judicious y adding and subtracting one (at the current bin's fineness) to the current bin's bynary latitude and longitudes

---

<sup>5</sup>That is, take the current location's latitude and longitude byns and use the current bin's finenesses to strip off (ignore) location data that is too fine for bins. Now generate nine byns (eight new and one that we started with) by adding -1, 0, +1 to the stripped location byns. Seek the nodes on the bin tree (discussed later) that correspond to these byns. If the byn corresponds to a child pointer of a node, and that child pointer points at a bin, search that bin. Otherwise, all bins pointed at by all of the descendants of that child pointer must be searched. If the byn does not correspond to any node on the tree, treat the byn as a location and search the bin that contains that location. In the last case, a bin should be searched only once, though it may be identified more than once.

The two byns used to define a terrain polygon or a bin do not have to have the same fineness. These *patches* can be rather trapezoidal, triangular, or even circular if close enough to the North or South Pole.<sup>6</sup> In fact, an area defined by lines of constant latitude and constant longitude is not, strictly speaking, a spherical polygon of any sort, as the parallels of latitude are not "straight lines" (that is, great circles). Almost everywhere, however, these distinctions are inconsequential and patches are virtually indistinguishable from ordinary rectangles.

The byns that define different bins do not have to have the same fineness. Longitude byns will often have smaller finenesses than latitude byns, because the length of a degree of longitude depends upon the latitude. At the equator, two 16-bit byns define a square patch 610.8 meters on a side. In Tokyo, at 36° N latitude, the patch defined by two 16-bit byns is a rectangle 610.8 by 495.25 meters.

Remote regions might have finenesses of seven bits (perhaps 100,000 square kilometers at the equator) or even fewer. On the other hand, a pair of fourteen-bit byns near the equator define an area about the same size as a three-kilometer hex.

How small can a patch defined by byns be? An equatorial patch specified by two binary coordinates, each of 24-bit fineness, has an area of only 5.7 square meters—about the same as a jeep. (A HMMV is larger; so is a tank; so are most units). Away from the equator, maximum fineness defines even smaller areas. At a pole, maximum fineness in both byns defines an area of only  $10^{-6}$  square meters (a triangle 2.386 meters in the N-S direction, 0.892 micrometers in E-W extent). There might be other applications that would require smaller bins, but not CBS.

### Locations and location objects.

In this paradigm, locations are areas, not mathematical points, but the areas implied by 24-bit byns are sufficiently small to be used as if they were point locations in the CBS context. Implementation of this idea does not, however, paint CBS into a corner. A few changes in low-level utility programs could improve the equatorial worst-possible resolution from 5.7 square meters to 0.54 square inches.<sup>7</sup> Surely *that* would satisfy any need we might ever have in CBS!

Binary latitude and longitude define a location for use inside CBS, but not for communications with users, who are familiar with the formats of a variety of other coordinate systems, such as conventional expressions of latitude and longitude, the U.S. Military Grid Reference System, the geographic Universal Transverse Mercator (UTM) system, and the World Geographic Reference

---

<sup>6</sup>A hunter leaves camp, travels 1 kilometer south, wanders  $2\pi$  kilometers east, shoots a bear, drags it 1 kilometer back to camp. What color is the bear?

<sup>7</sup>The finest resolution achievable with 24 bits is 0.07725 seconds of arc, producing a maximum location uncertainty of 5.7 square meters. Locations could be defined by all 32 bits, with 32-bit fineness assumed (rather than specified). The cost would be that locations and bins would not be defined and interpreted in the same ways. The benefit would be that locations could be 256 times as precise (to within 9.3 mm). Those low level routines could know the two uses apart when parsing. Since we have deemed, for convenience, that 8 bits define the fineness, while only 5 bits are actually needed, one of the extra bits could be used as a flag. The value 0.54 square inches assumes 31-bit byns. Another alternative would be to give up the efficiency resulting from storing the fineness in the byn and use additional words for fineness. As is often the case, trades between running time, storage requirements, and programming time are available.

System. For a while, it may even be necessary and desirable to translate to and from the pre-CBS 1.5.3 hex coordinates associated with a location.

CBS should translate from its native format into a user-friendly format whenever it is about to communicate with the user—but only then. It should not be necessary to update every variant way of expressing location data whenever a unit moves. One implementation would be to treat any location as an object, whose only data attributes are the binary latitude and longitude. Other formats, such as UTM, can be produced by function attributes only when they have to be displayed.

### **Sharing data.**

It is inevitable that many terrain patches will be described by identical values of all terrain parameters, except for elevation, because there are fewer combinations of possibilities than there are patches of terrain. Therefore, the large number of patches can be defined efficiently, with only a few attributes: pointers to *profiles* that contain the terrain data.

The complexity of these terrain profiles is a downstream implementation detail. Each polygon might have but one pointer, for example, which would point at one of  $9 \times 4 \times 3 \times \dots$  possible combination of terrain types. Alternatively, there might be a pointer to one of 9 possible combinations of troops, trucks, tracks x go, slowgo, nogo terrain; another pointer to one of 4 possible urbanization indices; another to one of 3 kinds of smoke; and so on. Each of these profiles might use lower-level data sharing.

The representations of roads and rivers in CBS 1.5.3 are also expected to use extensive data sharing. In that context, for example, many successive links in a road may share characteristics such as size, flow, condition, and perhaps even congestion.

Altitude data is likely to require special treatment, as we expect it to consist of values on a regular grid, with data spaced at perhaps 500 meter intervals. How many altitude points are in a bin? Should the entire bin be considered to be at the average height of those points? Should the bin maintain a list of altitude "poles", each of which has a location and an altitude? The strategy currently favored does not put the altitude data into the bins at all. Instead, altitude data is placed into a matrix with known, uniform spacings. Matrix indices can be readily determined from locations and vice versa.

### **Given a location, find the right bin and the right terrain data.**

Although it is still premature to fully design the data structure to be used, certain options immediately present themselves, due to the "double binary" nature of the scheme. For example, Eric Taylor has suggested we consider a tree constructed by sequentially dividing the world into quarters, some of which are divided further. Q When the patches defined during this process reach

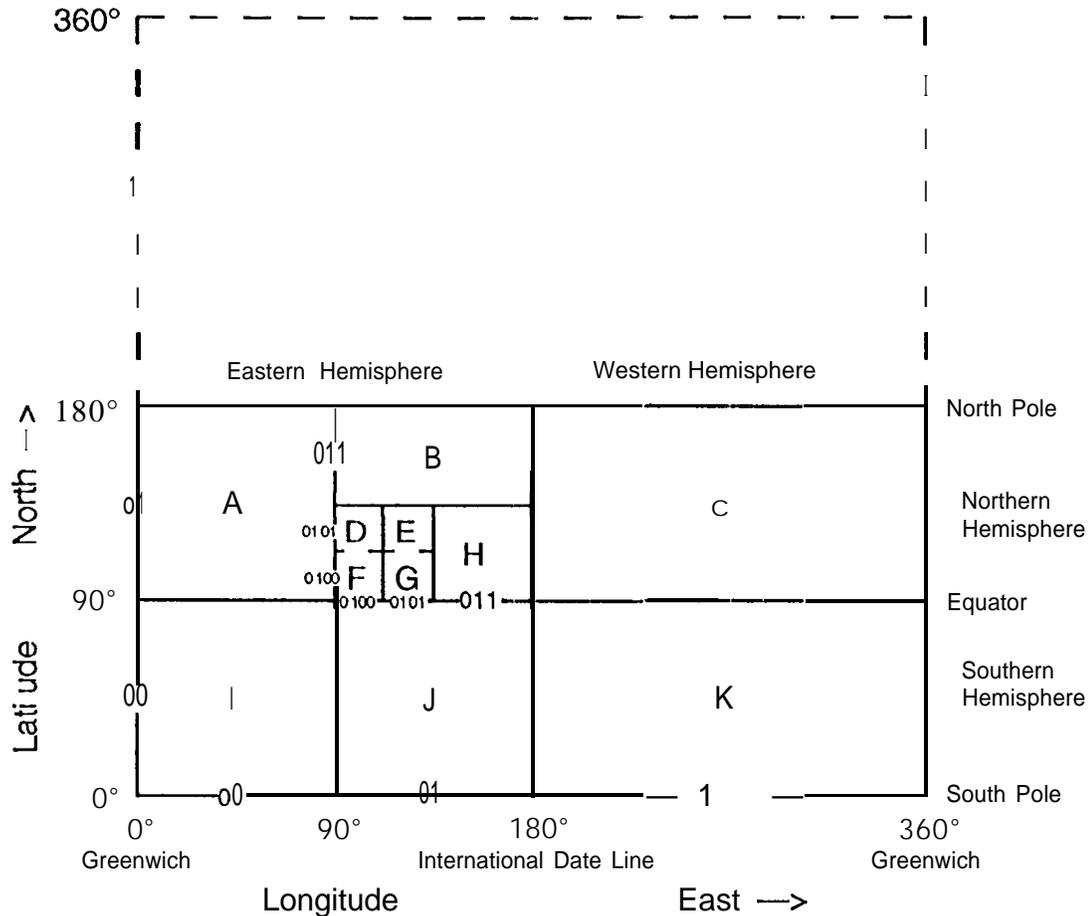
---

<sup>8</sup>The angular spacing between successive entries should be the same throughout the table, though not necessarily the same for latitudes as for longitudes. Due to the effect of latitude on the length of a degree of longitude, very large playboxes, or playboxes that include regions near the poles, may dictate the use of latitude-dependent matrices.

<sup>9</sup>Four quarters are produced by considering the possible combinations of halving both the latitude and longitude simultaneously. (There is an example in the nrxt paragraph.)

the finenesses chosen for bins, stop the process of division. Nodes on this tree have binary coordinates and four pointers to child nodes or bins.

Suppose the earth is to be represented by bins **A** through **K** as illustrated below. The corresponding tree is given on the following page.<sup>10</sup>

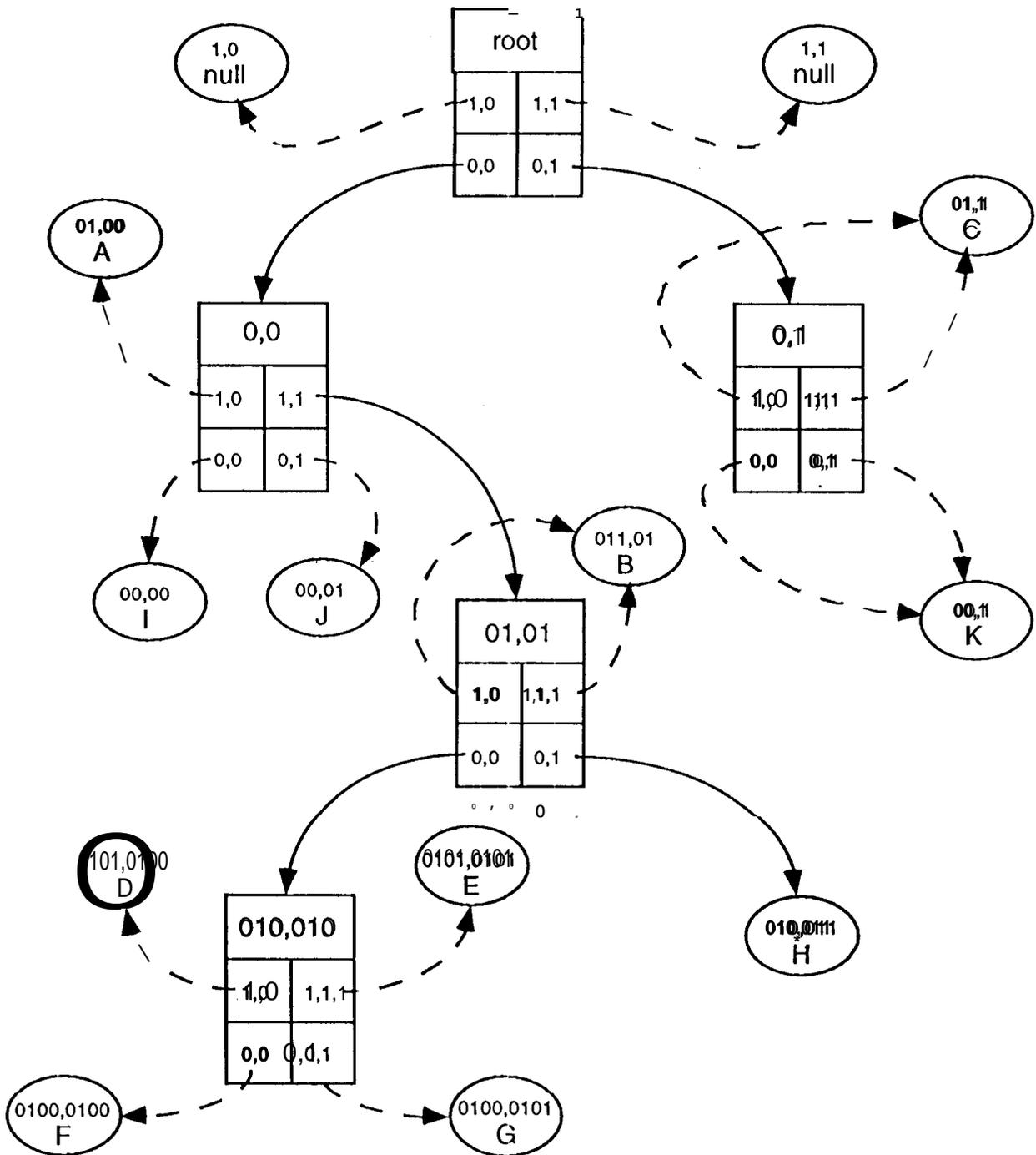


Bins have binary coordinates and contain lists of terrain patches, units, road and river network nodes and links, and other simulation objects that are located in the bin.

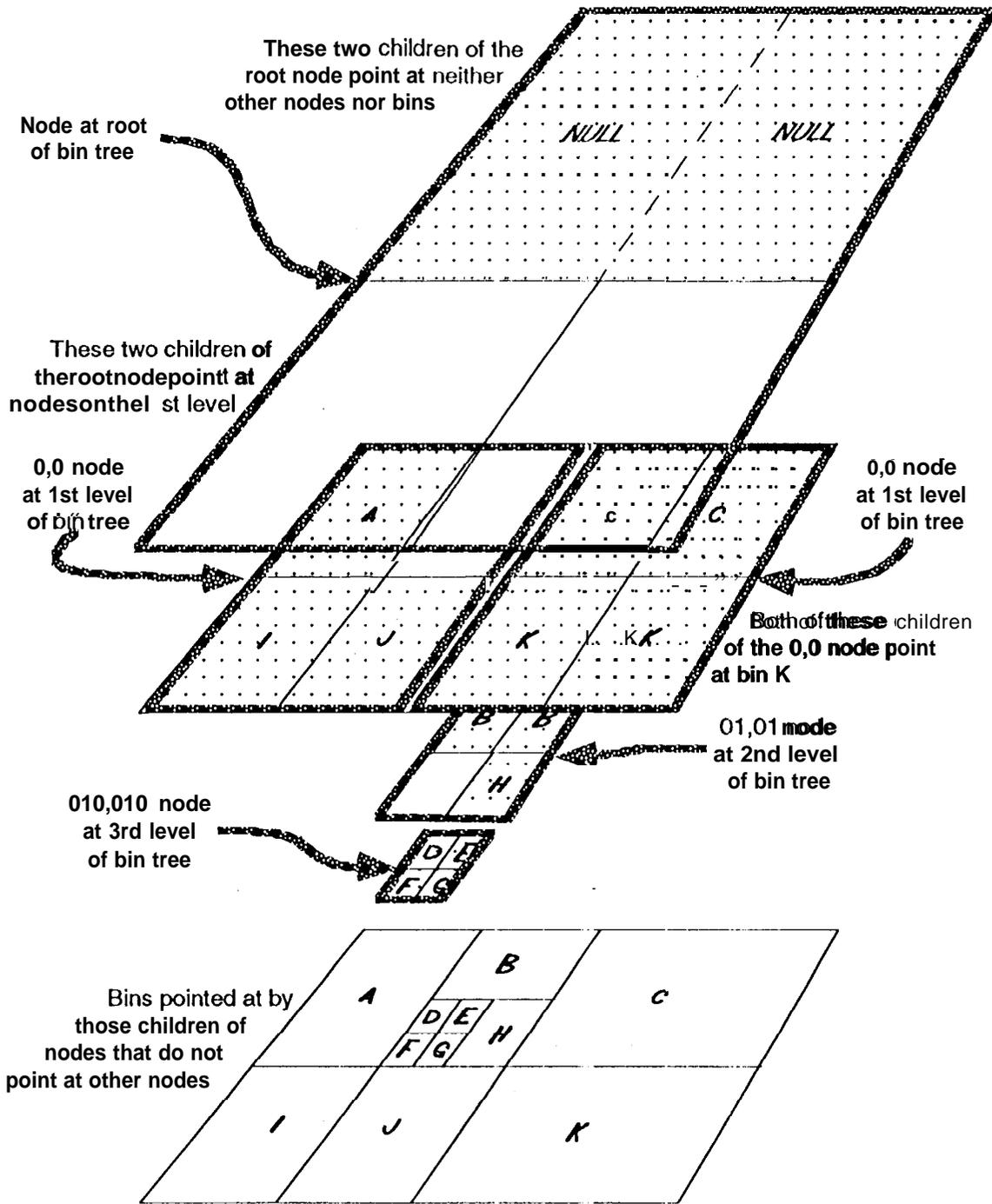
The binary coordinates of terrain patches and locations of simulation objects are produced by the same byning process as was used to develop the bin tree, but these other objects are not bins. Terrain patches have pointers to terrain profiles. Such other objects as minefield and antitank ditches also might use profiles to share data.

Suppose a unit is at coordinates 010001110,010111000. What bin is it in and what mobility conditions will the unit find? The first pair of bits (0,0) directs us to the node labeled 0,0. The next pair of bits (1,1) lead to node 01,01. Then to node 010,010. The 0,1 child of this node is bin G at 0100,0101.

<sup>10</sup>Finenesses are expressed by the number of binary digits shown; explicit representation has been suppressed in an attempt to clarify the illustration.



Structure of the bin tree. Nodes on the tree are represented by rectangles. The child/bin pointers are shown in relation to how they quarter the area represented by the binary coordinates of the node. Bins are represented by ovals containing the coordinates and the bin's "name" (A–K). An alternative representation is shown on the next page.



This also shows the structure of the bin tree, with an emphasis on the fact that the child pointers of nodes sometimes point at another node (shown as transparent) and sometimes at a bin (shown as opaque and dotted). Note the geometric proximity of the descendants of any particular node. Nodes are shown as fuzzy-edged oblique squares.

Now we know that the location is in bin G. That's all we need if we want to tell bin G that it should add something to its lists of data. To determine the mobility conditions, we look through the lists of mobility patches associated with the bin until we find one with the right coordinates, then look up the mobility profiles we find (perhaps "tracks slowgo, trucks nogo, troops go"). Similar data structures can be used to organize the data stored in bins.

### Spherical trigonometry: The distance between two points.

Question: "If locations are all expressed in spherical coordinate, am I going to have to use spherical trigonometry all the time?"

Answer: "No. Only when it matters."

Notation:

$\lambda_A$  = Latitude of location A, measured from the South Pole toward the North Pole (*not* measured from the equator, as is conventional), expressed in radians.

$\lambda_B$  = Latitude of location B, measured from the South Pole, expressed in radians.

$\Delta\lambda$  =  $\lambda_B - \lambda_A$  = Difference in latitudes between locations A and B, expressed in radians.

$\Delta\phi$  = Difference in longitudes between locations A and B, expressed in radians.

$R_E$  = Radius of the earth, approximately 6371 kilometers.

$d$  = Distance between locations A and B, expressed in radians, measured on a great circle.  
(To get the distance in kilometers, multiply  $d$  by  $R_E$ )

*Method 1. Not near either pole. Difference in longitudes small enough to ignore curvature of the parallels of latitude.* This will be the most common case. Angular differences of 0.07 radians (about 4°) or less ensure a relative error in the computed distance of less than one part in a thousand. This condition is met by separations of 500 km or less in the N-S direction and 100 km or less in the E-W direction anywhere except within the Arctic or Antarctic circles.

$$d^2 = \Delta\lambda^2 + \sin \lambda_A \sin \lambda_B \Delta\phi^2$$

This is the Pythagorean Theorem, though it may not look like it due to the effect of latitude on the circumference of a parallel of latitude (that is, the "length of a degree of longitude"). The product of the two sines, by the way, appears as a result of using Maclaurin series expansions for the cosines of the small angles  $d$ ,  $\Delta\lambda$ , and  $\Delta\phi$ .

*Method 2. Both locations close enough to the same pole to ignore curvature of the meridians. Any difference in longitudes.* Close to the poles, latitude and longitude constitute polar coordinates. (The terminology is not coincidental.) When both locations are within 500 km of the North or South Pole, the relative error in the distance computed by the following formula is sure to be less than one part in a thousand.

$$d^2 = \begin{cases} (\pi - \lambda_A)^2 + (\pi - \lambda_B)^2 - 2(\pi - \lambda_A)(\pi - \lambda_B) \cos \Delta\phi & \text{near the North Pole} \\ \lambda_A^2 + \lambda_B^2 - 2\lambda_A \lambda_B \cos \Delta\phi & \text{near the South Pole} \end{cases}$$

This expression is simply the Law of Cosines. An equivalent statement is

$$d^2 = \begin{array}{ll} \Delta\lambda^2 + 2(\lambda_A - \lambda_B)(1 - \cos\Delta\phi) & \text{near the North Pole} \\ \Delta\lambda^2 + 2\lambda_A\lambda_B(1 - \cos\Delta\phi) & \text{near the South Pole} \end{array}$$

**Method 3. The last resort. (The general case.) Difference in longitudes large enough to force consideration of the curvature of the parallels of latitude or difference in latitude large enough to force consideration of the curvature of the meridians. Any difference in longitude, any difference in latitude.**

$$d = \arccos(\cos\lambda_A \cos\lambda_B + \sin\lambda_A \sin\lambda_B \cos\Delta\phi)$$

Since we have defined the South Pole as the origin for latitudes, this formula maybe unfamiliar. Recall that the sine of an angle is the cosine of the complementary angle and vice versa. An equivalent statement which requires an extra subtraction but one less cosine evaluation is

$$d = \arccos(\cos\Delta\lambda - \sin\lambda_A \sin\lambda_B(1 - \cos\Delta\phi))$$

### Spherical trigonometry: Converting between linear and angular distances.

Given a latitude,  $\lambda_A$ , and differences in latitude and longitude,  $\Delta\lambda$  and  $\Delta\phi$ , what are the east-west and north-south distances, E and N?

$$E = \Delta\phi \cdot R_E \cdot \sin\lambda_A$$

$$N = \Delta\lambda \cdot R_E$$

Given a latitude,  $\lambda_A$ , east-west and north-south distances, E and N, what are the differences in latitude and longitude,  $\Delta\lambda$  and  $\Delta\phi$ ?

$$\Delta\lambda = N/R_E$$

$$\Delta\phi = E/(R_E \cdot \sin\lambda_A)$$

If  $\lambda_A \approx \pi$ , then  $\sin\lambda_A \approx 0.0$  and the bear must be white.

	$\lambda_A$	$\Delta\lambda$	$\Delta\phi$	Distance, km		
				Pythagoras (Method 1)	Polar (Method 2)	Exact (Method 3)
1° (111 km) South of the North Pole	179.0°	0°	1°	1.941	1.941	1.941
		0'	10°	19.406	19.382	19.381
		0'	90°	174.650	157.248	157.244
		-5°	90°	701.269	676.350	676.317
		-178°	0°	<b>19 792.045</b>	<b>19 792.045</b>	19792.045
Arctic Circle	156.5°	0°	1°	44.337	45.605	44.337
		0'	10°	443.374	455.475	442.901
		0'	90°	3990.367	3695.332	3641.983
		-5°	90°	4400.365	4107.309	4036.245
		5°	90°	3602.755	3325.530	3289.006
		-155°	0°	<b>11 234 .646</b>	<b>17 234.646</b>	17234.646
Oslo, Norway	149.9°	0°	1°	55.764	58.413	55.763
		0°	10°	557.636	583.396	557.106
		0'	90°	5018.725	4733.171	4618.942
		-5°	90°	5402.577	5141.343	4997.147
		5°	90°	4649.070	4357.818	4272.198
		-148°	0°	<b>16 456 .307</b>	<b>1 6 4 5 6 . 3 0 7</b>	16456.307
Los Angeles	124.1°	0°	1°	92.073	108.481	92.073
		0°	10°	920.731	1083.449	920.363
		v	90°	8286.577	8790.174	7970.242
		-5°	90°	8530.365	9191.705	8247.887
		5°	90°	8041.340	8406.251	7704.828
		-1 20°	0°	<b>13 342.951</b>	<b>13 342.951</b>	13342.951
Quito, Ecuador	89.8°	v	1°	111.191	174.463	111.191
		0'	10°	1111.911	1742.434	1 111.911
		0'	90°	10007.198	14136.612	10007.194
		-5°	90°	10002.839	13749.113	10006.225
		5°	90°	10004.362	14535.050	10008.163
		90°	0°	10007.214	10007.214	10007.214