

DISCRETE GLOBAL GRID SYSTEMS: A NEW CLASS OF
GEOSPATIAL DATA STRUCTURES

by

KEVIN MICHAEL SAHR

A DISSERTATION

Presented to the Department of Computer
and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

August 2005



DISCRETE GLOBAL GRID SYSTEMS: A NEW CLASS OF
GEOSPATIAL DATA STRUCTURES

by


KEVIN MICHAEL SAHR

A DISSERTATION

Presented to the Department of Computer
and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

August 2005

“Discrete Global Grid Systems: A New Class of Geospatial Data Structures,” a dissertation prepared by Kevin Michael Sahr in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Department of Computer and Information Science. This dissertation has been approved and accepted by:



Dr. John Conery, Chair of the Examining Committee



Date

Committee in Charge: Dr. John Conery, Chair
 Dr. Andrzej Proskurowski
 Dr. Arthur Farley
 Dr. Patrick Bartlein

Accepted by:



Dean of the Graduate School


© 2005 Kevin Michael Sahr

An Abstract of the Dissertation of

Kevin Michael Sahr for the degree of Doctor of Philosophy
in the Department of Computer and Information Science to be taken August 2005

Title: DISCRETE GLOBAL GRID SYSTEMS: A NEW CLASS OF GEOSPATIAL
DATA STRUCTURES

Approved: _____



Dr. John Conery

Limitations in traditional approaches to the representation of geo-referenced data sets has led to the development of a number of data structures based on regular, multi-resolution partitions of spherical polyhedra. These constitute a new class of geospatial data structures that we call Discrete Global Grid Systems (DGGs). After defining an abstract data type for structured geospatial data structures that encompasses DGGs we survey the proposed DGG approaches. We show that the primary DGG alternatives can be constructed by specifying five substantially independent design choices: a base regular polyhedron, a fixed orientation of the base regular polyhedron relative to the Earth, a hierarchical spatial partitioning method defined symmetrically on a set of faces of the base regular polyhedron, a method for transforming that planar partition to the corresponding spherical/ellipsoidal surface, and a method for assigning point representations to grid cells. An examination of the design choice options leads us to the construction of the Icosahedral Snyder Equal Area aperture 3 Hexagon (ISEA3H) DGGs.

We next develop a topology-independent implementation of DGGs based on our abstract data type that will enable us to perform empirical comparisons between the primary DGG topologies of hexagons, triangles, and diamonds. Since a major advantage of DGGs is that they can function as topologies for dynamic simulation and analysis, for our initial comparison we implement a simple dynamic simulation by extending the definition of a planar cellular automata to be spherical, multi-scale, and topology-independent. We then report the first results for a study of such simulations.

Finally, we note that the practical use of icosahedral aperture 3 DGGs, such as the ISEA3H, has been hindered by a lack of efficient hierarchical location coding schemes. We introduce two path-based hierarchical location coding systems: the Icosahedral Modified Generalized Balanced Ternary approach for indexing point data, and the Icosahedral Aperture 3 Hexagon Tree for indexing raster data and for use in bucket-based spatial databases. Algorithms for conversion from geographic coordinates to these systems are given.

This dissertation includes both my previously published and my co-authored materials.

CURRICULUM VITAE

NAME OF AUTHOR: Kevin M. Sahr

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon
University of Colorado at Colorado Springs
Bucknell University

DEGREES AWARDED:

Doctor of Philosophy in Computer and Information Science, 2005,
University of Oregon
Master of Science in Computer Science, 1995, University of Colorado
at Colorado Springs
Bachelor of Arts in Mathematics and Philosophy, 1984, Bucknell University

AREAS OF SPECIAL INTEREST:

Spatial Data Structures
Computer Graphics

PROFESSIONAL EXPERIENCE:

Assistant Professor, Southern Oregon University, 2000 – present
Graduate Teaching Fellow, University of Oregon, 1997 – 2000
Research Associate, Oregon State University, 1993 – 1999
Software Engineer, CAE-Link Corporation, 1992 – 1993
Programmer/Analyst, Kaman Sciences Corporation, 1989 - 1992

GRANTS, AWARDS AND HONORS:

Research Grant from GRIDS, Limited, 2003

Distinguished Military Graduate, 1984

Army ROTC four-year scholarship recipient, 1980

PUBLICATIONS:

Sahr, K. (2004, October). *Spatial indexing on icosahedral aperture 3 discrete global grids*. Paper presented at the Second International Conference on Discrete Global Grids, Ashland, OR.

Kiester, R., & Sahr, K. (2004, October). *Spatially hierarchical cellular automata on discrete global grids*. Paper presented at the Second International Conference on Discrete Global Grids, Ashland, OR.

Kimerling, A.J., & Sahr, K. (2004, October). *Using data loss and duplication maps as tools for comparing data use when resampling from equal-angle, hexagonal, and triangular discrete global grids*. Paper presented at the Second International Conference on Discrete Global Grids, Ashland, OR.

Gregory, M., Kimerling, J., White, D., & Sahr, K. (2004, October). *Evaluating desirable geometric characteristics of discrete global grid systems: Revisiting the Goodchild criteria*. Paper presented at the Second International Conference on Discrete Global Grids, Ashland, OR.

Sahr, K. (2004, March). *Developing software for discrete global grids*. Paper presented at GeoTec 2004, Toronto, Canada.

Sahr, K., White, D., & Kimerling, A.J. (2003). Geodesic discrete global grid systems. *Cartography and Geographic Information Science*, 30(2), 121-134.

Song, L., Kimerling, A.J., & Sahr, K. (2002). Developing an equal area global grid by small circle subdivision. In M.F. Goodchild & A.J. Kimerling (Eds.), *Discrete global grids: A web book*. Santa Barbara, CA: University of California. Retrieved June 1, 2005, from <http://www.ncgia.ucsb.edu/globalgrids-book/song-kimmerling-sahr>

Sahr, K. (2000, March). *A preliminary comparison of proposed topologies for geodesic discrete global grid systems*. Paper presented at the First International Conference on Discrete Global Grids, Santa Barbara, CA.

Gregory, M., Kimerling, J., White, D., & Sahr, K. (2000, March). *Comparing intercell distance and cell wall midpoint criteria for discrete global grid*

systems. Paper presented at the First International Conference on Discrete Global Grids, Santa Barbara, CA.

- Kiester, R., & Sahr, K. (2000, March). *Unexpected and complex behavior of hierarchical, multiresolution cellular automata*. Paper presented at the First International Conference on Discrete Global Grids, Santa Barbara, CA.
- Kimerling, A.J., Sahr, K., White, D., & Song, L. (1999). Comparing geometrical properties of global grids. *Cartography and Geographic Information Science*, 26(4), 271-287.
- Sahr, K., & White, D. (1998). Discrete global grid systems. In S. Weisberg (ed.), *Computing science and statistics (volume 30): Proceedings of the 30th Symposium on the interface, computing science and statistics* (pp. 269-278). Fairfax Station, VA.: Interface Foundation of North America.
- White, D., Kimerling, A. J., Sahr, K. & Song, L. (1998). Comparing area and shape distortion on polyhedral-based recursive partitions of the sphere. *International Journal of Geographical Information Science*, 12, 805-827.
- Carr, D.B., Kahn, R., Sahr, K., & Olsen, A.R. (1997). ISEA discrete global grids. *Statistical Computing & Graphics Newsletter*, 8(2/3), 31-39.
- Kimerling, A. J., Sahr, K., & White, D. (1997). Global scale data model comparison. *Proceedings of Auto-Carto 13: American Congress on Surveying and Mapping*, 357-366.
- Arthur, J. L., Hachey, M., Sahr, K., Huso, M., & Kiester, A.R. (1997). Finding all optimal solutions to the reserve site selection problem: Formulation and computational analysis. *Environmental and Ecological Statistics*, 4, 153-165.
- Csuti, B., Kennelly, P., Meyers, S.M., & Sahr, K. (1997, July). *Current status of biodiversity indicators using GIS*. Paper presented at the 1997 ESRI User Conference, San Diego, CA.
- Csuti, B., Polasky, S., Williams, P.H., Pressey, R.L., Camm, J.D., Kershaw, M., Kiester, A.R., Downs, B., Hamilton, R., Huso, M., & Sahr, K. (1997). A comparison of reserve selection algorithms using data on terrestrial vertebrates in Oregon. *Biological Conservation*, 80, 83-97.
- Kiester A.R., Scott, J.M., Csuti, B., Noss, R., Butterfield, B., Sahr, K., & White, D. (1996). Conservation prioritization using GAP data. *Conservation Biology*, 10(5), 1332-1342.

ACKNOWLEDGMENTS

Over a decade ago I had the privilege to begin working with a very special group of people, at what came to be called the Terra Cognita Laboratory in the Department of Geosciences at Oregon State University. It is only through the support of all the faculty, scientists, graduate students, and administrators associated with Terra Cognita over the years that this research has come to fruition, and to all of them I am most grateful. In particular, I wish to give a special thanks to Ross Kiester, Jon Kimerling, Tony Olsen, and Denis White for their many-varied forms of support over the years.

In addition, I wish to thank my advisor, Dr. John Conery, and the department graduate secretary, Star Holmberg, for all their assistance in shepherding me through this process in spite of myself.

Finally, I want to thank another very special group of people without whose support I would never have finished: my wife Carolyn and my children Stephen, Matika, Hope, and Natalia.

To Ross Kiestler, for always finding the right problems.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION AND BACKGROUND	1
Motivation	1
Background: Geospatial Data Structures	4
Fundamental Geospatial Operation Classes	7
Traditional Geospatial Data Structures	12
The Next Step	23
II. GEODESIC DISCRETE GLOBAL GRID SYSTEMS.....	24
Acknowledgements	24
Discrete Global Grid Systems: Basic Definitions	24
Geodesic Discrete Global Grid Systems	28
Summary and Conclusions	43
Directions for Further Research	48
III. NULIB: A TOPOLOGY-INDEPENDENT DGGS ARCHITECTURE	50
Motivation	50
Implementing the Nulib Core	51
Applications of Nulib	58
IV. SPATIALLY HIERARCHICAL CELLULAR AUTOMATA ON DISCRETE GLOBAL GRIDS	61
Acknowledgements	61
Introduction	61
Extending Cellular Automata to DGGSs.....	62
Results 65	
Conclusions	70
V. LOCATION CODING ON ICOSAHEDRAL APERTURE 3 HEXAGON DISCRETE GLOBAL GRIDS	71
Introduction	71
Background	72
Pyramid Addressing on Aperture 3 Hexagon Grids: The Quadrilateral Two-Dimensional Integer System	75
Path Addressing on Aperture 3 Hexagon Grids	81
Quantization Algorithm for the iA3HT	92
Conclusions	102

Chapter	Page
VI. SUMMARY AND CONCLUSIONS	106
BIBLIOGRAPHY	109

LIST OF FIGURES

Figure	Page
1. Two Definitions of Adjacency on Square Rasters	15
2. Two Definitions of Adjacency on Triangle Rasters	15
3. Hexagon Grids Display Uniform Adjacency.....	16
4. A Portion of Two Resolutions.....	27
5. Planar and Spherical Versions of the Five Platonic Solids	30
6. Spherical Icosahedron Orientation.....	32
7. Three Levels of a Class I Aperture 4 Triangle Hierarchy	34
8. Three Levels of a Class II Aperture 3 Triangle Hierarchy.....	34
9. Three Levels of a Aperture 4 Diamond Hierarchy.....	36
10. Seven-Fold Hexagon Aggregation	37
11. Three Levels of a Class I Aperture 4 Hexagon Hierarchy	39
12. Three Levels of a Class II Aperture 4 Hexagon Hierarchy.....	39
13. Three Levels of an Aperture 3 Hexagon Hierarchy.....	39
14. Three Resolutions of Icosahedron-Based Geodesic DGGs	40
15. ETOPO5 5' Global Elevation Data Binned into the ISEA3H.....	48
16. Hierarchical Adjacency Set for the Planar Aperture 4 Triangle Topology	56
17. Hierarchical Adjacency Set for the Planar Aperture 4 Diamond Topology.....	56
18. Hierarchical Adjacency Set for the Planar Aperture 3 Hexagon Topology	57
19. Hierarchical Adjacency Set for the Planar Aperture 4 Hexagon Topology	57
20. Four-Resolution Aperture 3 Hexagon Grid Initialized	65
21. Sample Time Series	68
22. ACF for Representative Hexagon Sample Series	69
23. Three-axis Hexagon Coordinate Systems	76
24. Class I 2di Coordinate System	77
25. Four Resolutions of an Aperture 3 Hexagon Grid System	78
26. Icosahedral Faces Paired to Form Ten Quadrilaterals	79
27. Unfolded Icosahedron with 2di Coordinate Systems	79
28. Class II Grid on a Single Quadrilateral	80

29. Quantization at One Resolution Restricts Possible Quantization Candidates	81
30. MGBT Addressing of Class II Children	82
31. MGBT Addressing of Class I Children	82
32. Sub-Regions Addressed by MGBT	83
33. iMGBT Base Tiling on an Unfolded Icosahedron	85
34. A3HT Open (Type A) Generator Hexagon	86
35. A3HT Closed (Type B) Generator Hexagon	87
36. Generation of Four Resolutions of an A3HT Grid	88
37. A3PT Open (Type A) Generator Unfolded on the Plane	89
38. A3PT Closed (Type B) Generator Unfolded on the Plane.....	90
39. iA3HT Base Tiling on an Unfolded Icosahedron	91
40. The First Six Resolutions of an iA3HT Generation Pattern	92
41. Relationship Between Base iA3HT Tiles and the Corresponding Q2di Quadrilateral	101

LIST OF TABLES

Table	Page
1. Minimal ADT Components of an RF.....	12
2. Primary Forms of the Proximity Operators	12
3. Summary of Geodesic DGGs Design Choices.....	45
4. Life State Transition Table	63
5. Generalized Life State Transition Table	64
6. Mean Number of Time Steps Achieved	66
7. Substitutions for Rotation of an A3HT Cell	99
8. Quadrilateral q vs. iA3HT Base Index A	100
9. iA3HT Base Cell Look-Up and Adjustments	103

CHAPTER I

Introduction and Background

Motivation

Many of today's most challenging computer science applications involve data referenced to the earth's surface. These include problems in global climate modeling, environmental monitoring and assessment, transportation planning, and military modeling and simulation. Integral to all of these applications is some representation of the geospatial extent of the data elements, which we will refer to as a *geospatial data structure*.

Increasingly inexpensive technologies (e.g., embedded Global Positioning System (GPS) receivers) allow for the real-time update of the location of mobile computing devices — and hence the location of the device's mobile user. This ability will greatly enhance the user's interface to the physical world by facilitating location-specific access to geospatial data (where is the closest pizza parlor?) and algorithms (what is the shortest route from here to New York City?) (Noronha, 2000). The new field of *location-specific* or *location-aware computing* has begun to develop to address the needs of these user communities.

Advances in computing hardware, GIS software, and in collection-related technologies such as satellites and GPS receivers have facilitated the availability of global spatial data sets of increasingly fine granularity, allowing researchers to analyze, visualize, and model processes on a global scale at increasingly higher resolutions. The familiar, convenient, and relatively transparent structure of the raster grid makes it the data structure of choice for high-resolution satellite imagery. But simple raster grids are not efficient in terms of storage usage, making such datasets among the most massive in

use. For example, NASA's multi-platform Earth Observation System generates over one terabyte of geospatially referenced raster data per day (Shekhar & Chawla, 2003). Effective use of such datasets will push the limits of current massive database technology.

Active, on-going research in the often overlapping fields of geographic information systems (GIS), spatial data structures and algorithms, spatial databases, and computer graphics has provided geospatial applications with a rich set of techniques, data structures, and algorithms (see, for example, the surveys in Samet, 1989, 1990; Shekhar, Chawla, Ravada, Fetterer, Liu, and Lu, 1999; Rigaux & Voisard, 2002; Shekhar & Chawla, 2003). Techniques developed to solve general problems in spatial computing have been adapted with a great deal of success to geospatial computing.

But these techniques primarily use approaches to the representation of geospatial location developed in the pre-computer days of paper-based mapping, in which locations on the earth's topologically spherical surface are projected onto a portion of the plane before mapping them into a geospatial data structure. Given the challenging nature of modern geospatial computing problems, and the prevalence of high-resolution global data sets, researchers have begun to search for more efficient approaches to representing geospatial location. Alternative data structures have been proposed based on highly regular, multi-resolution partitions of polyhedral approximations of the earth's surface. Sahr, White, and Kimerling (2003) have proposed the name *Geodesic Discrete Global Grid Systems* (Geodesic DGGS) for this class of spatial data structures.

Geodesic grids have been proposed for a wide variety of geospatial applications, including global climate modeling (e.g., Thurn, 1997), storage of continuous field values (Fekete & Treinish, 1990), the development of statistically sound survey sampling designs (White, Kimerling, and Overton, 1992; Olsen, Stevens, and White, 1998), and even as a replacement for traditional geospatial coordinate systems (Dutton, 1989). The Quaternary Triangular Mesh (QTM) system (Dutton, 1999) has been used in the development of a number of global geospatial algorithms including optimal path planning (Stefanakis & Kavouras, 1995), line simplification (Dutton, 1999), and as an index for spatial databases (Dutton, 1999). QTM has been used as the basis for spatial indexing

systems in commercial applications such as Microsoft Encarta (Dutton, 1999). The popularity of QTM, and other DGGS based on 4-fold recursive decomposition of triangles or squares (e.g., Alborzi & Samet (2000)), seems in large part due to the fact that they induce data structures which are forests of quadtrees. The quadtree is a well-studied class of spatial data structure, allowing these DGGS to be quickly exploited using traditional models developed for planar geospatial data structures.

But GIS researchers and geospatial data end-users have proposed DGGS based on alternative tilings, such as the hexagon. Studies by GIS researchers (Kimerling, Sahr, White, and Song, 1999) and mathematicians (Saff & Kuijlaars, 1997) indicate that hexagon-based DGGS may have clear advantages for many applications. A hexagon-based DGGS has been adopted by the US Environmental Protection Agency for global sampling problems (White et al. 1992). And hexagon-based DGGS have been proposed at least 5 times in the atmospheric modeling literature (Williamson, 1968; Sadournay, Arakawa, and Mintz, 1968; Heikes & Randall, 1995a, 1995b; Thuburn, 1997; Ringler & Heikes, 1999), to our knowledge more often than any other Geodesic DGGS topology.

While many DGGS alternatives have been known to the spatial data structures community for over a decade (see, for example, Samet's (1989, 1990) classic two-volume survey of spatial data structures), designs based on topologies other than the 4-fold decomposition of triangles have received little attention from data structures researchers. Multi-resolution hexagon grids, in particular, do not fit well with current models of hierarchical geospatial data structures. The cells of such grids cannot be created by simple aggregation of atomic pixels, nor by recursive partition. Such grids do not induce hierarchical data structures which are quadtrees, or even strictly trees. But hexagon-based DGGS are clearly an important approach that cannot be ignored by the geospatial data structures community. To quote the conclusion of Sahr et al. (2003): "a significant effort must be made by the data structures community to develop and evaluate algorithms for the regular, but non-tree, hierarchies they form."

If we wish to objectively evaluate the relative potential of all DGGS alternatives as geospatial data structures, then we must begin with a model that includes them all. Our

goal, then, is to develop a theoretical framework for the definition of global geospatial data structures that can serve both as the basis for a denotational semantics of DGGs, facilitating analytical comparisons, and as a basis for the implementation of these alternatives, making possible empirical evaluations. In the next section we will attempt to define an abstract data type for geospatial data structures, based upon a survey of the requirements that have conditioned the representation of geospatial location. In the next chapter we will survey the major proposed DGGs alternatives. In subsequent chapters we will describe our implementation of the abstract data type developed here to provide a test platform for comparing discrete simulations defined independent of particular DGGs topologies, and use this implementation to implement an initial comparison experiment using a multi-resolution, topology independent spatial simulation. Finally, we will use the terminology we have developed to describe efficient hierarchical location coding methods for one of the most promising hexagon-based DGGs alternatives, the Icosahedral Snyder Equal Area Aperture 3 Hexagon (ISEA3H) DGGs. In the final section we discuss what we believe to be key research issues remaining as we move forward in developing DGGs as practical geospatial data structures.

Background: Geospatial Data Structures

Any attempt to define new geospatial data structures should begin by defining an abstract data type (ADT) for the effective representation of geospatial location. An ADT can serve as the basis for a denotational semantics of DGGs, facilitating analytical comparisons, and as a basis for the implementation of these alternatives, making possible empirical evaluations. An ADT must include a description of the basic form of the data elements, and of the operations that must be supported. Such a model must be inclusive of the requirements that geospatial applications impose upon geospatial data structures. But we must be clear to distinguish between those requirements that are inherent to geospatial computing, and those that have been imposed by traditional geospatial data structures and practice.

Geospatial computing applications involve an association between data objects and a representation of some portion of the earth's surface. This representation is sometimes referred to as the *spatial* or *geometric extent*. We will call such a representation the *location* of the data object. The basic data element is then the two-tuple [*data-object*, *location*], though this may be implemented in a variety of equivalent forms (e.g., the location may be represented parametrically and calculated as needed). Data objects range from a single value (e.g., the strength of a continuous field, such as surface temperature), to an arbitrarily complex representation of some object on the earth's surface (e.g., an agent representing a person in an individual-based military simulation). In practice the location associated with a data object may represent any of a number of *location forms*: a zero-dimensional point, a one-dimensional line or curve, a two-dimensional region, or an arbitrary set of these primitive location forms.

A point associated with a data object may represent the actual location of the object if its spatial extent is a point (as in the case of the point strength of a continuous field) or it may be the centroid, or some other significant point, associated with a higher dimensional spatial extent. A region associated with an object may be the exact region that the object occupies, or it may be a simplified region, often a convex bounding region. Sometimes both representations are maintained, and the simplified representation can be used as a coarse filter to increase the efficiency of some geometric operations such as equality, intersection, or containment. For example, an object intersection algorithm can test for intersection between two locations by first testing for intersection between the simplified bounding regions; this can greatly reduce the number of more expensive intersection tests amongst full location representations. Perhaps the most common simplified region representation is a minimum bounding axes-aligned rectangle, the smallest bounding rectangle with sides parallel to the orthogonal axes of a two-dimensional cartesian coordinate system. Such a representation allows for particularly simple and efficient algorithm definition.

Locations are defined in the context of a particular geospatial data structure, or *reference frame (RF)*. An RF defines the possible values for location representations, as

well as the set of operations defined on these representations. We call an RF-specific location representation an *address*. A location, then, is a two-tuple $[RF, address]$. We designate that the location p is defined as an address in the RF α (i.e., $p = [\alpha, address]$) by writing p^α .

Whether the address is fully materialized and stored, or computed as needed using some spatially referenced function, on current digital computers it must be expressible as a string of bits. Let A^α be the set of all addresses a^α for some RF α . Note that by definition A^α may be its own power set and theoretically infinite in cardinality. But for any particular instantiation of α on a digital computer there exists some finite number of bits n available for address representation; therefore $|A^\alpha|$ is finite and less than 2^n . If α supports a point location form we may define P^α , the set of all point location addresses p^α . Note that P^α is a subset of A^α . Let n_α designate the *base resolution* of α , defined $n_\alpha = \log|P^\alpha|$.

Casati and Varzi (1996) distinguish between location representations which are *structured* (e.g., $145.2^\circ W$ longitude, $45^\circ N$ latitude) and those that are *unstructured* (e.g., *Poland* and *Warsaw*). Structured RFs are those that can be used to specify or calculate arbitrary spatial relationships. Spatial relationships among addresses in unstructured RFs must be specified explicitly, or calculated using quantitative operations on corresponding locations defined on a structured RF. An RF on which a limited set of relations have been defined is known as a *semi-structured* representation. The set of relations in a semi-structured RF are often represented as a directed graph.

For example, *Poland* and *Warsaw* are both meaningful addresses in the unstructured RF *List-of-Places*. A semi-structured RF that defined mereological (part/whole) relations (e.g., “*Warsaw is-part-of Poland*”) could efficiently implement useful mereological inference operations. But such a data structure cannot be constructed based solely on the unstructured RF *List-of-Places*. We could, however, calculate these relationships by testing for intersections between polygons representing the boundaries of each place, defined as polygons in the structured geographic (*latitude, longitude*) RF. Semi-structured RFs are thus fundamentally dependent on structured RFs; they support only a

specific set of operations directly and must rely on a structured RF to provide general purpose geospatial computing support, either storing or calculating on-the-fly the required structured RF object locations. Semi-structured RFs cannot, in and of themselves, meet the requirements of a general purpose global geospatial location system.

Our geospatial ADT must be compatible with the diverse set of structured and semi-structured RFs in use by geospatial applications; if our ultimate goal is a general-purpose geospatial ADT, we cannot restrict ourselves to RFs favored by any particular class of application. Indeed, by definition an ADT for structured geospatial data structures must support *any* operation which has been or might be defined on an RF. Simply compiling a list of the many operations required by current geospatial applications would, at the very least, create an unwieldy ADT. We must instead attempt to identify fundamental operations from which these application-specific operations can be composed, ideally determining a minimal set of core operations. By definition these would constitute a minimal set of operations necessary to define an ADT for structured RFs. In the next section we will discuss some classes of geospatial operators that appear to be fundamental to any structured RF. In the following section we will survey traditional geospatial data structures to determine the viability of our ADT definition, as well as to understand the kinds of location address forms that are currently in use.

Fundamental Geospatial Operation Classes

An ideal general-purpose RF would faithfully mirror all aspects of the rich geometric structure of our shared experience of physical geospatial location. To quote Nievergelt (1989) “the first point to be made about spatial data structures sounds so trite that it is often overlooked: The main task of a spatial data structure is to represent space.” The main task of an RF, then, is to represent the spatial structure of the surface of the earth. The surface of the earth is thus the *primary RF*, which we designate as *E*. The fundamental nature of RFs as representations of *E* implies that every RF α must define a mapping between addresses in *E* and addresses in α . If operations in α produce new loca-

tions then the inverse operation, mapping addresses in α to addresses in E , must also be defined for those locations to be meaningful.

Let *quantization* be the operation of mapping an address in one RF to an address in another. Formally, given two RFs α and β the quantization operator $\Rightarrow^{\alpha\beta}: A^\alpha \rightarrow A^\beta$ defines a location p^β for each location p^α . For convenience we will often write a particular application of this operator to a location p^α as $p^\alpha \Rightarrow p^\beta$. Note that in general a series of quantization operations $\Rightarrow^{\alpha\beta}, \Rightarrow^{\alpha\gamma}$ can be composed into a single operation $\Rightarrow^{\alpha\gamma}$, though such a chained quantization operation may not constitute a unique operation; i.e., there may be multiple quantization paths between any two RFs.

From the above we can conclude that every RF α must define a *primary quantization operator* $\Rightarrow^{E\alpha}: A^E \rightarrow A^\alpha$ and will usually define the *inverse primary quantization operator* $\Rightarrow^{\alpha E}: A^\alpha \rightarrow A^E$. In the case of unstructured RFs these operations may be defined by direct association. For example, the address *Poland* in the unstructured RF *List-of-Places* is known to correspond to a particular bounded region that can be indicated directly on E . Alternately, the operations can be defined to/from an RF with a well-defined correspondence to E , such as a geographic RF with a fully specified datum; note that this RF would usually be structured. In such cases we can compose a primary quantization operator $\Rightarrow^{E\alpha}$ for an RF α from the operations $\Rightarrow^{E\beta}$ and $\Rightarrow^{\beta\alpha}$, where β is the structured RF surrogate for E . If there exists multiple quantization paths between E and α we can, without loss of generalization, choose one of them to be the primary quantization operator.

The primary quantization operator determines the basic efficiency and accuracy of representation for the RF. It is the one operation that must be defined for any RF, whether unstructured, semi-structured, or structured. Associated with each act of quantization is some (possibly zero) *quantization error*, which we will define explicitly later in the context of specific RFs. It is important to distinguish between quantization operations between RFs that share the same base partition (i.e., which have a one-to-one correspondence between address values) and those which do not. We call the former *partition-equivalent* RFs. For now we note that we can define quantization operators between parti-

tion-equivalent RFs which have zero quantization error, while this is not in general the case for RFs which are not partition-equivalent.

In addition to the quantization operator, a number of other operations are generally accepted as fundamental to any structured RF. Probably the most fundamental of these are *equality* operators.

The base partition of an RF forms the basis for the fundamental operation *isE-qualTo* $^\alpha: A^\alpha \times A^\alpha \rightarrow bool$. Given an RF α two locations p_1 and p_2 are considered equal in α if and only if they have the same address in α (i.e., the address components of p_1^α and p_2^α are elements of the same location equivalence class in A^α); this is written $p_1^\alpha == p_2^\alpha$. Usually equality operations are only defined between equivalent location forms (point-point, curve-curve, region-region, and set-set). In some data structures locations may be considered equal if they are “very close” (see the definition of distance operators below and the specific example in the next section), but this is really a redefinition of the semantics of the abstract data type in response to difficulties with a particular data structure.

Ideally the operations which we define are invariant with respect to quantization. A predicate relation $operator^\alpha: A^\alpha \times A^\alpha \rightarrow bool$ has this property if and only if

$$operator^\alpha(p_1^\alpha, p_2^\alpha) == operator^E(\Rightarrow^{\alpha E}(p_1^E), \Rightarrow^{\alpha E}(p_2^E))$$

for all locations p_1 and p_2 . Given locations defined in different RFs, we can apply quantization operators to one or both of the locations to transform them into a common RF for equality testing. Note, however, that the choice of common RF may affect the result unless all of the RFs involved are partition equivalent.

Equality is one of a class of fundamental geospatial operators which we call *proximity* operators; these are operations that involve the notion of nearness and farness, and capture the concept of coincidence of location. These operations form a natural progression of increasing generalization that is loosely analogous to degree of proximity. Listed in order from high to low proximity they are: *containment*, *equality*, *intersection*, *adja-*

cency, and *distance*.

The containment predicate operator $isContainedIn^\alpha: A^\alpha \times A^\alpha \rightarrow bool$ is true if and only if the first argument is a proper subset of the second argument. Containment operations are defined between equivalent location forms. In addition, a location may be contained within a location form of higher dimension (e.g., a point may be contained in a line or a region).

In addition to predicate forms, proximity operations also have *generation* and *selection* forms. For example, the generation containment operator $contains^\alpha: A^\alpha \rightarrow A^\alpha$ generates the set of all locations that are contained in the specified location argument. Let D be the set of all data objects associated with the current application. Then the selection containment operator $selectContainedIn^\alpha: A^\alpha \rightarrow D$ would select all data objects in D for which the associated location is contained in the indicated location argument.

The intersection predicate operator $isIntersectedBy^\alpha: A^\alpha \times A^\alpha \rightarrow bool$ is true if and only if the intersection of the two location arguments is non-empty. Intersection operations are well-defined between any pair of location forms, though they are equivalent to containment operations when one location is of higher dimension, and to an equality operation when both are points.

Distance (ideally metric) operations have an unambiguous definition only in the case of point locations. Given an RF α we can define the graph G^α with P^α as the set of vertices and with a unit-length edge defined between each pair of addresses in P^α if and only if the actual geospatial regions associated with these two addresses are physically “next to” each other on the surface of the earth (this will be discussed more fully in the context of specific data structures in the next section). Then we can define the value of the operation $distance^\alpha(p_1^\alpha, p_2^\alpha)$ to be the length of the shortest path between p_1^α and p_2^α in G^α . Note that by definition the distance operator returns a positive (or zero) integer result; the distance operator has the form $distance^\alpha: A^\alpha \times A^\alpha \rightarrow \mathbb{Z}^+$.

Equality and adjacency (or *neighborhood*) operations can be defined as special cases of the distance operation. The equality predicate $isEqualTo^\alpha(p_1^\alpha, p_2^\alpha)$ is true if and only if $distance^\alpha(p_1^\alpha, p_2^\alpha)$ is zero. The adjacency predicate $isNextTo^\alpha(p_1^\alpha, p_2^\alpha)$ is true if

and only if $distance^\alpha(p_1^\alpha, p_2^\alpha)$ is one.

Distance operators also allow us to quantify the *absolute error* introduced by a quantization operation as the distance between the original and quantized locations, defined in the original RF. That is, the absolute error associated with a specific quantization operation $p^\alpha \Rightarrow p^\beta$ is given by $distance^\alpha(p^\alpha, \Rightarrow^{\beta\alpha}(p^\beta))$. The *characteristic error* of a quantization operator $\Rightarrow^{\alpha\beta}$ is the maximum absolute error for all locations in P^α .

Quantization and proximity operators appear to have a unique status in the literature as the absolute minimum set of operations required for a structured RF. While perhaps not as widely mentioned, one additional class of operators is often also considered more fundamental than others in capturing a notion of location: the *direction* operators. These operators attempt to capture information about the relative position of locations beyond the notion of proximity. Directions are most commonly defined between point locations and are commonly represented as either vectors or angles. Let M^α be the set of directions defined on the RF α . Then two common direction operators are $directionTo^\alpha: P^\alpha \times P^\alpha \rightarrow M^\alpha$, which calculates the direction from one location to another, and $directionOffset^\alpha: P^\alpha \times M^\alpha \times Z \rightarrow P^\alpha$, which calculates the location a given distance in a given direction from an initial point.

To summarize, then, we may conclude that the minimal structured geospatial ADT defines an RF α as the set of sets $\{A^\alpha, P^\alpha, G^\alpha, M^\alpha\}$ with the following operations defined: a primary quantization operator, proximity operators (*containment, equality, intersection, adjacency, and distance*), and direction operators (*directionTo* and *directionOffset*). Table 1 summarizes the ADT set components and Table 2 summarizes the primary forms of the proximity operators.

Table 1. Minimal ADT components of an RF α .

component	description
A^α	set of addresses in α
P^α	set of point locations in A^α
G^α	graph induced by P^α
M^α	set of directions defined on α
D	set of data objects

Table 2. Primary forms of the proximity operators.

operation	predicate form	generation form	selection form
containment	$isContainedIn^\alpha:$ $A^\alpha \times A^\alpha \rightarrow bool$	$contains^\alpha: A^\alpha \rightarrow A^\alpha$	$selectContainedIn^\alpha:$ $A^\alpha \rightarrow D$
equality	$isEqualTo^\alpha:$ $A^\alpha \times A^\alpha \rightarrow bool$	$equalTo^\alpha: A^\alpha \rightarrow A^\alpha$	$selectEqualTo^\alpha:$ $A^\alpha \rightarrow D$
intersection	$isIntersectedBy^\alpha:$ $A^\alpha \times A^\alpha \rightarrow bool$	$intersects^\alpha: A^\alpha \rightarrow A^\alpha$	$selectIntersects^\alpha:$ $A^\alpha \rightarrow D$
adjacency	$isNextTo^\alpha:$ $A^\alpha \times A^\alpha \rightarrow bool$	$neighbors^\alpha: A^\alpha \rightarrow A^\alpha$	$selectNeighbors^\alpha:$ $A^\alpha \rightarrow D$
distance	$isDistanceTo^\alpha:$ $A^\alpha \times A^\alpha \times Z^+ \rightarrow bool$	$neighborhood^\alpha:$ $A^\alpha \times Z^+ \rightarrow A^\alpha$	$selectNeighborhood^\alpha:$ $A^\alpha \times Z^+ \rightarrow D$

Traditional Geospatial Data Structures

Now that we have identified a minimal set of fundamental operations for an abstract data type for the representation of geospatial location, let us turn our attention to the specific forms that RFs have traditionally taken. This will allow us to validate our set

of fundamental operators, to identify additional operators for our ADT, and to understand the basic forms that have been used for geospatial data structures in the past. For this purpose it is useful to distinguish three broad geospatial application domains: the *end-user domain*, the *spatial data model domain*, and the *resource mapping domain*. We will discuss each of these application domains in detail below. But briefly, end-user domain applications are those in which an RF is chosen which transparently mimics the geospatial representation form used by the end-user in formulating their problem. In spatial data model domain applications the end-user location representations are quantized into RFs chosen specifically to facilitate efficient algorithms for important operations. Resource mapping domain applications are those in which locations are explicitly mapped to physical resources (or vice versa), and RFs are chosen to optimize resource usage.

While a number of data structures and algorithms may be applied to more than one of these domains, each domain is characterized by a distinct emphasis in requirements that tend to condition the form of the RFs employed under that domain. We will discuss each of these in turn. While we believe our approach and organization is original, most of the material itself follows traditional lines (see in particular Samet, 1990; Rigaux et al. 2002; Shekhar & Chawla, 2003).

The End-User Domain

When end-users formulate geospatially referenced problems they typically use some preferred mathematical representation of geospatial location. When translating these problems into computer models the end-user often finds it convenient to choose an RF which mimics their preferred mathematical representation, and that supports similar operations. We term this class of geospatial data structure usage the *end-user domain*. There are traditionally two distinct approaches to the representation of location by end-users, often referred to as *raster* and *vector* approaches. Both approaches are considered structured. We will discuss each of these in turn.

Under the raster approach to location representation the spatial domain of interest is partitioned by a uniform grid. Usually the raster grid is not truly uniform on the earth's surface, but is defined on some mathematical space that acts as a surrogate representation of that surface. Common surrogate spaces include planar surfaces produced by the application of map projections to portions of the earth's surface, cylindrical projections of the entire earth surface, and polar coordinate spaces defined on a spherical or ellipsoidal approximation of the earth's surface, which produces the widely used latitude/longitude graticule.

Raster RFs map conveniently onto the abstract data type given above; a raster RF α is specified by assigning each grid cell a unique address. The address graph G^α has a convenient geometric analogue which can be constructed by placing a uniform lattice of vertices on the surrogate surface and then connecting adjacent vertices with edges. The spacing of these vertices determines the *resolution* of the raster, or the minimum size of details which the raster can distinguish. The faces (or voronoi cells) of the graph vertices form the location equivalence classes. Each graph face is called a *pixel*. Quantization of point locations is performed in a geometrically intuitive manner by mapping each point on the surrogate surface to the nearest graph vertex, with some rule specified for the assignment of points falling on the boundary between cells (commonly referred to as "rounding"). Curves and regions are represented by the minimal set of pixels which they intersect, contain, or are contained by, depending upon the application.

The most common raster form is generated by a regular square point lattice with square pixels. It is favored because it is the traditional basis for spatial grids among end-users, and it allows for extremely efficient quantization to/from traditional cartesian coordinate systems. The most common pixel address form is a two-tuple of integers, allowing for a simple two-dimensional matrix representation of the domain space that is highly transparent to most generic end-users. Because each square pixel has four neighbors with which it shares an edge, and another four neighbors with which it shares only a vertex, there are two possible definitions for adjacency. As illustrated in Figure 1, under the $D4$ metric only edge neighbors are considered adjacent, while under the $D8$ metric both edge

and vertex neighbors are considered adjacent. Both lead to definitions for the distance operator which are metrics.

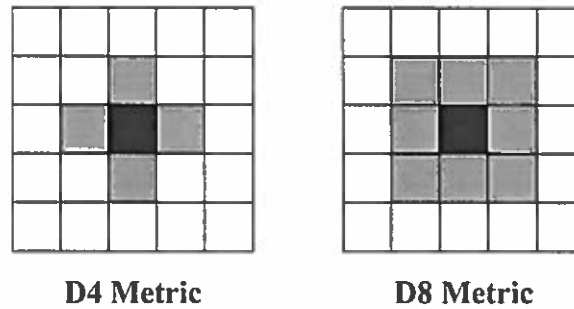


Figure 1. Two definitions of adjacency on square rasters generate two different metrics.

In addition to the square pixel raster, a number of other pixel shapes have been used. The most uniform shapes, and therefore those which have received the most attention for raster RFs (see, for instance, the classic and influential study Bell, Diaz, Holroyd, and Jackson, 1983), are regular polygons that tile the plane. There are three such shapes: the square, the regular hexagon, and the equilateral triangle.

As with square grids, triangle grids have two possible definitions for adjacency, depending on whether or not vertex neighbors of each cell are considered adjacent. As illustrated in Figure 2, this leads to two possible metric definitions. In addition, triangle grid pixels do not have uniform orientation; each pixel can point up or down. Thus the orientation of each pixel must be taken into account when developing algorithms on triangle grids.

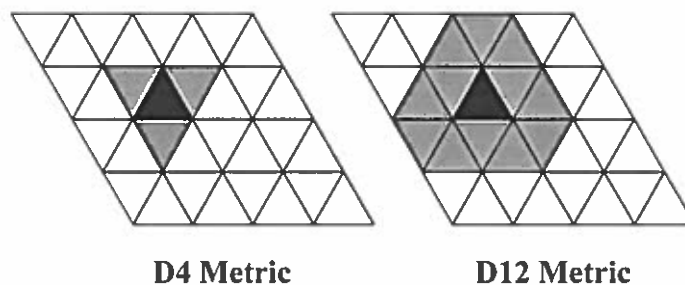


Figure 2. Two definitions of adjacency on triangle rasters generate two different metrics.

Hexagon grids have received a great deal of attention. Among the three regular polygons that tile the plane, hexagons are the most compact, they quantize the plane with the smallest average error (Conway & Sloane, 1988), and they provide the greatest angular resolution (Golay, 1969). Unlike square and triangle rasters, hexagon rasters do have uniform adjacency. As illustrated in Figure 3, each hexagon pixel has six neighbors, all of which share an edge with it, and all of which have centers exactly the same distance away from its center. Each hexagon pixel has no neighbors with which it shares only a vertex. This fact alone has made hexagons increasingly popular as bases for discrete spatial simulations. Frisch, Hasslacher, and Pomeau (1986) argue that the six discrete velocity vectors of the hexagonal lattice are necessary and sufficient to simulate continuous, isotropic, fluid flow. A recent textbook (Rothman & Zaleski, 1997) on fluid flow cellular automata is based entirely on hexagonal meshes, with discussions of square meshes included “only for pedagogical calculations.” Triangle grids, which are even more limited for this purpose, are not mentioned.

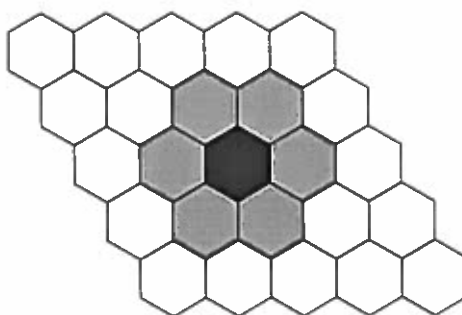


Figure 3. Hexagon grids display uniform adjacency.

However, hexagon rasters have a disadvantage in that hierarchical grids can not be built of hexagons using the usual techniques of aggregating small hexagons into larger hexagons or, conversely, sub-dividing large hexagons into smaller ones.

The second major end-user domain approach to location representation is the vector approach, in which locations are given a mathematical representation. There are two common types of vector representation. Under the first, point locations are represented by

assigning them some coordinate value, usually in a cartesian coordinate system. A line segment is represented by specifying its endpoints, and arbitrary curves are approximated by a series of connected line segments (a *polyline*), and represented as the ordered vector of point locations (*nodes* or *vertices*) connected by the line segments (*arcs*). Similarly, in this vector approach regions are represented using a closed polyline (a *polygon*) constituting their boundary. The second common type of vector location representation is to indicate each location form using a parametric equation. Note that even when this second approach is employed, particular materialized location address representations are almost always given in some coordinate system, as under the first approach. The most common address representation for points under the vector approach is a two- or three-dimensional vector of real numbers. Traditionally these are either polar (latitude/longitude) coordinates or cartesian coordinates defined in some planar map projection space. Such locations are conveniently represented as two-tuples of floating point numbers.

Both raster and vector RFs can be constructed using a lattice of fixed points; in this sense their underlying mathematical structure is identical. The difference between them is in the way in which they are perceived and utilized. Raster RFs are sometimes called *space-* or *field-based* models, because in such data structures the underlying space itself is the primary organizing principle. Often a fully materialized matrix forms a data structure analog to the geospatial area of interest, with data objects associated with corresponding matrix locations. In contrast, vector RFs are *entity-* or *object-based* models, with the data objects being the primary, materialized, components with which particular location addresses are associated.

This reflects an underlying assumption that vector RFs are somehow more like real number coordinate systems than the explicitly discrete raster RFs; indeed vector addresses are sometimes referred to as “exact” addresses. Thus operations are conveniently defined on the vector RF addresses to mimic the corresponding mathematical operations defined on vectors of real numbers. But while the cartesian plane is continuous, representations of cartesian coordinates on digital computers are always finite and discrete. Operations defined on these finite addresses can introduce rounding errors into

the resulting location addresses. Consequently even fundamental operations on vector RFs must take into account the possibility of propagated rounding error. For example, earlier we defined the equality predicate to be true if and only if the distance between the two address arguments is zero. But in the case of vector addresses they are usually considered equal if the distance between them is less than some relatively small number. This makes it impossible to distinguish between two addresses which are distinct yet very close, and two addresses which are intended to indicate the same location on the earth's surface but which differ due to rounding error when one or both of them were calculated. In many applications this distinction is of little consequence and hence can be ignored without effect. However, in the case of spatial location applications the result of a location equality test may well be an important decision point in determining the future course of execution for the application. And while it is often possible to bound the rounding errors in the calculations of a single application run, modern geospatial computing increasingly involves interactions between multiple geospatial applications. In such situations it may be very difficult to bound the overall round-off error in the final system results.

In addition to the fundamental operators of quantization, proximity, and direction, end-user applications make use of a diverse variety of operations. These include the common constructive solid geometry operations (*union*, *exclusive-or*, etc.), affine transforms (*translation*, *rotation*, and *scaling*), and measurement operations such as *length* and *area*. Both raster and vector RFs are commonly used as spatial substrates for stochastic simulations which require, in addition to the fundamental operations already listed, operations that introduce a stochastic element such as *movement* (change in address value over time) and *deformation* (arbitrary change in location representation associated with a data object over time).

Because they are explicitly finite, raster grids also support *traversal* operations, or spatially explicit orderings of all location addresses (and hence of all associated data objects). Additionally, raster grids are widely used for storing and processing digital imagery, and a diverse set of *map algebra* operations have been defined for this purpose.

These operations transform pixel values based on linear (and sometimes non-linear) functions of the original pixel values. Examples of such operations include *thresholding*, *frequency filtering*, and *edge enhancement*. Note that the fundamental adjacency and distance operations are used in the definition of many map algebra operations.

The GIS applications domain deserves special mention. Applications in this area are distinguished by the emphasis they place on RF location, in and of itself, as a primary object of analysis, manipulation, and visualization. As throughout the end-user domain, vector and raster RFs are the primary RFs used in GIS applications. But the GIS community straddles the boundary between the end-user domain and the spatial data model domain discussed in the next section. GIS applications include all of the operations so far discussed, though the community has developed its own terminology for many of these structures and operations. For example, the *topological model* is the GIS term for the representation of basic location forms as points, lines represented as connected series of points (*polylines*), and regions represented by closed polyline boundaries. GIS, in conjunction with the fields of computational geometry and computer graphics, has developed a rich set of algorithms for efficiently representing and manipulating location representations such as the topological model. For example, representational redundancy in a topological model can be reduced by introducing a vertex- or edge-table representation.

The Spatial Data Model Domain

More computationally sophisticated users are much less concerned with addressing transparency than they are with the efficiency with which an RF can be used to implement specific sets of operations. We refer to this user domain as the spatial data model domain. Research in the computer science data structures community has traditionally focussed on this domain, with a particular emphasis on two areas: *spatial graph models* and *spatial databases*.

Geospatial graph or network models are traditional graphs in which the nodes rep-

resent distinct locations and the edges represent connectivity between these locations; note that by definition graph models are semi-structured RFs. The graph edges may be weighted. In one common example, the nodes represent cities and the edges represent highways connecting the cities, with edge weights indicating the length of each route. In some cases the edges are directed, as when the graph represents drainage paths in a river system.

Most of the traditional graph operations have direct applications to geospatial graphs. For example, *shortest path determination* and *graph traversal* operations have important application in transportation planning. Indeed, the oft-cited traveling salesman problem (Lawler, Lenstra, Kan, and Shmoys, 1985) is explicitly a geospatial graph problem.

The spatial database domain is concerned with the efficient storage and retrieval of geospatially referenced data. Spatial database research attempts to integrate spatial data with existing relational database technology as well as to develop new approaches tailored to the unique requirements of spatial database users. Spatial data differs from traditional data records in a number of respects. In most data sets fields used as index keys have limited mathematical structure relevant to query construction; this structure is usually limited to a linear ordering. In spatial databases the location address is a primary key, and as we have seen these addresses have a rich mathematical structure. This structure is routinely exploited in building spatial database queries. Traditionally these queries have been limited to the fundamental proximity operators listed above. But recently it has become clear that users desire the ability to build queries using arbitrarily complex spatial operators; in the case of relational databases these are known as *arbitrary spatial joins*.

Both structured and semi-structured RFs have been used as the basis for spatial databases. Since structured RFs support arbitrary spatial operations by definition they allow for the construction of arbitrary spatial queries. The *quadtree* is the basis for a variety of hierarchical structured RFs developed for storing various location forms. It is an efficient RF for storing field-based data like raster grids. However, the quadtree can be

very inefficient for storing entity-based or sparse field-based data objects. Some researchers have proposed modified quadtree structures that use non-uniform subdivision adjusted to the distribution of data objects in space. But such adjustments degrade the regular structure that is the quadtree's chief advantage.

The cells of the multi-resolution quadtree can be used as buckets for storing data objects, with data objects associated with the highest resolution cell that entirely contains them, known as the *minimum bounding cell*. This tree of minimum bounding cells can then be queried using spatial operators. However, in most cases the tree itself will not be height-balanced, resulting in inefficient search algorithms. Non-spatial databases with linear keys make use of height-balanced tree structures such as the *B-tree* to increase search and data access efficiency. As previously noted, linear traversal orderings can be defined on raster RFs, and such orderings can then form the basis of a B-tree structure. However, since the geospatial surface is usually two-dimensional the transformation to one-dimension destroys the proximity structure of the original raster. Guttman (1984) proposed the *R-tree* semi-structured data structure as a natural two-dimensional extension of the B-tree. Object locations are represented in this structure by minimum bounding rectangles. While this creates an efficient height-balanced tree for geospatial data, it has at least two limitations. First, the minimum bounding rectangles are only useful as a coarse filter for proximity queries; the actual object locations must be stored in the data structure so that they are available for exact query computations. This is an example of the fact we discussed earlier, that semi-structured RFs are always dependent on structured RFs to provide general purpose spatial support. The other difficulty introduced by the R-tree is that the minimum bounding rectangles are not disjoint. This can lead to inefficient processing of some spatial queries. A number of R-tree variants have been proposed to increase the query performance of these structures, sometimes at the expense of more restrictive assumptions about the data set.

The spatial database problem is directly concerned with the mapping of spatial data to secondary storage and thus it straddles the boundary between the spatial data model domain and the resource mapping domain discussed in the next section.

The Resource Mapping Domain

The resource mapping domain involves geospatial applications where the primary criteria is the efficiency of the mapping between geospatial data and physical resources. There are two basic problem classes in this domain: the efficient mapping of geospatial RFs to computational resources, and the use of computations on geospatial RFs to drive the efficient distribution of location-dependent physical resources on the surface of the earth.

The primary computational resource mapping problem has traditionally been the problem of efficiently mapping non-linear (usually two-dimensional) spatial surfaces to linear (one-dimensional) secondary storage. This has traditionally been solved using one of two approaches: explicit linearization of the RF (with the consequence of loss of proximity), or by decomposing the space into buckets that are then mapped to storage units; for example, each leaf node of an R-tree usually corresponds to a single page on disk. The rise of distributed and massively parallel computing architectures, and the desire to apply these architectures to complex geospatial problems such as global climate modeling, has led to a more general problem of mapping subsets of geospatial RFs to arbitrary computing nodes connected with diverse communications topologies, while attempting to retain the original geospatial proximity. Algorithms have also been developed which take advantage of structured hierarchical RFs like the quadtree. For instance, traditional field-based integration algorithms make iterative use of neighbor data values to constrain pixel values; *multi-grid* algorithms incorporate constraints between parents/children within the hierarchy and often result in faster convergence.

An increasingly important geospatial application area is the calculation of efficient distributions of geospatially constrained physical resources on the earth's surface; perhaps the most studied such problem over the last decade has been the allocation of limited-range cell phone transmitters to provide for optimal coverage. In the fast-growing field of location-based computing, it has long been recognized that location-specific data repositories (and associated physical resources) are most efficiently located near the

associated location on the earth's surface. The optimal distribution of these resources and their associated communication topologies for efficient access by mobile users presents new challenges in geospatial computing.

The Next Step

Our survey of geospatial computing applications has yielded a set of fundamental abstract data type operations and an overview of the data structures that have been developed to meet the needs of geospatial computing applications. As previously noted, a number of researchers have found limitations in these traditional approaches when applied to the increasingly common high-resolution global data sets. Extensions of these approaches onto topologically spherical data structures has led to a new class of structured geospatial data structures known as DGGs. In the next chapter we will survey the primary DGG alternatives that have been proposed.

CHAPTER II

Geodesic Discrete Global Grid Systems

Acknowledgements

This chapter is substantially the same as the previously published and co-authored paper Sahr et al. (2003). The present author was the primary developer of the definitions and classification scheme that constitute the main contributions of that paper. Figure 6 was created by Denis White. All other figures were created by the present author.

Discrete Global Grid Systems: Basic Definitions

Discrete Global Grid

A Discrete Global Grid (DGG) consists of a set of *cells*, where each cell is a tuple consisting of an areal region on the surface of the earth, and a single point contained in that region. The cell regions form a partition of the Earth's surface. Depending on the application, data objects or vectors of values may be associated with regions, points, or cells. If an application defines only the regions, the centroids of the regions form a suitable set of associated points. Conversely, if an application defines only the points, the Voronoi regions of those points form an obvious set of associated cell regions.

Applications often use DGGs with cell regions that are irregular in shape and/or size. For example, the division of the Earth's surface into land masses and bodies of water constitutes one of the most important DGGs. A more general example, the Hipparchus System (Lukatella, 2002), allows the creation of arbitrarily regular DGGs by generating Voronoi cells on the surface of an ellipsoid from a specified set of points. But for many applications it is desirable to have highly regular cells, with regions as uniform

in shape and size as possible, and with cell points evenly distributed across the surface of the earth.

Regular DGGs are unbiased with respect to spatial patterns created by natural and human processes and allow for the development of simple and efficient algorithms. A single regular DGG may play multiple data structure roles. It may function as a raster data structure, where each cell region constitutes a pixel. It may serve as a vector data structure, where the set of DGG points replaces traditional coordinate pairs (Dutton, 1999). Each data object may be associated with the smallest cell region in which it is fully contained, and these minimum bounding cells may then be used as a coarse filter in operations such as object intersection. The DGG can also be used as a useful graph data structure by taking the DGG points as the graph vertices and then connecting points associated with neighboring cells with unitweight edges.

The most commonly used regular DGGs are those based on the geographic (latitude–longitude) coordinate system. Raster global data sets often employ cell regions with edges defined by arcs of equal-angle increments of latitude and longitude (for example, the $2.5^\circ \times 2.5^\circ$, $5^\circ \times 5^\circ$, and $10^\circ \times 10^\circ$ NASA Earth Radiation Budget Experiment [ERBE] grids described in Brooks, 1981). Data values may also be associated with points spaced at equal-angle intervals of latitude and longitude (for example, the 5' \times 5' spacing of the ETOPO5 global elevation data set described in Hastings & Dunbar, 1998). Similarly, vector data sets that employ a geographic coordinate system to define location values commonly choose a specific precision for those values. The choice of a specific precision for geographic coordinates forms an implicit grid of fixed points at regular angular increments of latitude and longitude, and a particular data set of that precision can consist only of coordinate values chosen from this set of fixed points. Ideally, the regions associated with the geographic vector coordinate points would be the corresponding Voronoi regions on the Earth's surface, although they are more commonly the corresponding Voronoi regions on a surrogate representation of the Earth's surface, such as a sphere or ellipsoid. In practice, applications often implicitly employ Voronoi regions defined on the *longitude \times latitude* plane, on which the regions are, conveniently, regular

planar squares. As we shall see, employing a surrogate representation for the Earth's surface on which the cell regions are regular planar polygons is a useful and common approach in DGG construction.

Discrete Global Grid System

A Discrete Global Grid System (DGGs) is a series of discrete global grids. Usually, this series consists of increasingly finer resolution grids; i.e., the grids in the series have a monotonically increasing number of cells. If the grids are defined consistently using regular planar polygons on a surrogate surface, we can define the *aperture* of a DGGs as the ratio of the areas of a planar polygon cell at resolution k and at resolution $k+1$ (this is a generalization of the definition given in Bell et al., 1983). Later we will discuss DGGs that have more than one type of polygonal cell region. In these cases there is always one cell type that clearly predominates, and the aperture of the system is defined using the dominant cell type. Kimerling et al. (1999) and Clarke (2002) note the importance of regular hierarchical relationships between DGGs resolutions in creating efficient data structures. Two types of hierarchical relationship are common. A DGGs is *congruent* if and only if each resolution k cell region consists of a union of resolution $k+1$ cell regions. A DGGs is *aligned* if and only if each resolution k cell point is also a cell point in resolution $k+1$. If a DGGs does not have these properties, the system is defined as *incongruent* or *unaligned*. For example, the widely used geographic (*latitude, longitude*) vector coordinate system constitutes a DGGs generated implicitly by multiple precisions of decimal geographic vector representations. This DGGs has an aperture of 10 and is incongruent and aligned (Figure 4).

Discrete Global Grid Systems based on the geographic coordinate system have numerous practical advantages. The geographic coordinate system has been used extensively since well before the computer era and is therefore the basis for a wide array of existing data sets, processing algorithms, and software. Grids based on square partitions are by far the most familiar to users, and they map efficiently to common data structures

and display devices. But such grids also have limitations. Discrete Global Grid Systems induced by the latitude-longitude graticule do not have equal-area cell regions, which complicates statistical analysis on these grids. The cells become increasingly distorted in area, shape, and inter-point spacing as one moves north and south from the equator. The north and south poles, both points on the surface of the globe, map to lines on the longitude x latitude plane; the top and bottom row of grid cells are, in fact, triangles, not squares as they appear on the plane. These polar singularities have forced applications such as global climate modeling to make use of special grids for the polar regions. Square grids in general do not exhibit uniform adjacency; that is, each square grid cell has four neighbors with which it shares an edge and whose centers are equidistant from its center. Each cell, however, also has four neighbors with which it shares only a vertex and whose centers are a different distance from its center than the distance to the centers of the edge neighbors. This complicates the use of these grids for such applications as discrete simulations.

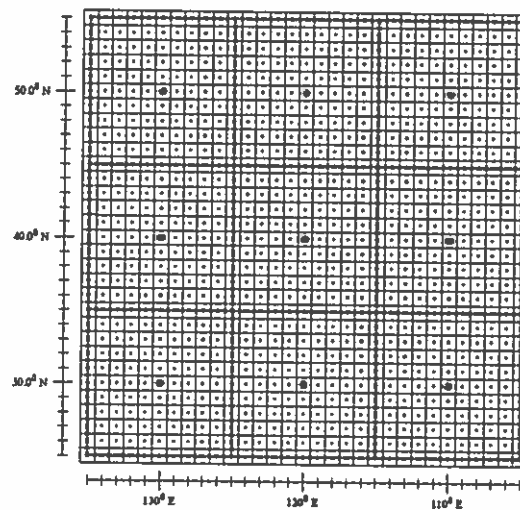


Figure 4. A portion of two resolutions (10° and 1° precision) of the DGGs implicitly generated by multiple precisions of decimal geographic coordinate system vector representations. Note that this is an incongruent, aligned hierarchy.

Attempts have been made to create DGGs based on the geographic coordinate system but adjusted to address some of these difficulties. For example, Kurihara (1965)

decreased the number of cells with increasing latitude so as to achieve more consistent cell region sizes. Bailey (1956), Paul (1973), and Brooks (1981) used similar adjustments of latitude and/or longitude cell edges to achieve cell regions with approximately equal areas. But these schemes achieved more regular cell region areas at the cost of more irregular cell region shapes and more complex cell adjacencies. Tobler and Chen (1986) projected the Earth onto a rectangle using a Lambert cylindrical equal area projection and then recursively subdivided that rectangle, but, as in the case of the other mentioned approaches, this did not address the basic problem that the sphere/ellipsoid and the plane are not topologically equivalent.

Geodesic Discrete Global Grid Systems

The inadequacies of DGGs based on the geographic coordinate system have led a number of researchers to explore alternative approaches. Many of these approaches involve the use of regular polyhedra as topologically equivalent surrogates for the Earth's surface, and, in our opinion, these attempts have led to the most promising known options for DGGs. A number of researchers have been inspired directly or indirectly by R. Buckminster Fuller's work in discretizing the sphere, which led to his development of the geodesic dome (Fuller, 1975). We will thus refer to this class of DGGs as Geodesic Discrete Global Grid Systems.

Geodesic DGGs have been proposed for a number of specific applications. Inherently regular in design, these systems have most commonly been used to store raster data sets, but they may also be used as a substitute for traditional coordinate-based vector data structures (Dutton, 1999), as data containers, or as the basis for graphs (as described in the previous section). Geodesic DGGs have been used to develop statistically sound survey sampling designs on the Earth's surface (Olsen et al., 1998), for optimum path determination (Stefanakis & Kavouras, 1995), for line simplification (Dutton, 1999), for indexing geospatial databases (Otoo & Zhu, 1993; Dutton, 1999; Alborzi & Samet, 2000), and for the generation of spherical Voronoi diagrams (Chen, Zhao, and

Li, 2003). They have also been proposed as the basis for dynamic simulations such as those used in global climate modeling (Williamson, 1968; Sadournay et al., 1968; Heikes & Randall, 1995a, 1995b; Thuburn, 1997).

It is highly unlikely that any single Geodesic DGGs will ever prove optimal for all applications. Many of the proposed systems include design innovations in particular areas, though their construction may have involved other, less desirable design choices. Therefore, rather than surveying individual Geodesic DGGs as monolithic, closed systems, we will take the approach here of viewing the construction of a Geodesic DGGs as a series of design choices which are, for the most part, independent. The following five design choices fully specify a Geodesic DGGs:

1. A base regular polyhedron;
2. A fixed orientation of the base regular polyhedron relative to the Earth;
3. A hierarchical spatial partitioning method defined symmetrically on a face (or set of faces) of the base regular polyhedron;
4. A method for transforming that planar partition to the corresponding spherical/ellipsoidal surface; and
5. A method for assigning points to grid cells.

We will now look at each of these design choices in turn, discussing the decisions made in the development of a number of Geodesic DGGs.

Base Regular Polyhedron

As White et al. (1992) and many others have observed, the spherical versions of the five platonic solids (Figure 5) represent the only ways in which the sphere can be partitioned into cells, each consisting of the same regular spherical polygon, with the same number of polygons meeting at each vertex. The platonic solids have thus been commonly used to construct Geodesic DGGs, although other regular polyhedra have sometimes been employed. Among these the truncated icosahedron has proved to be pop-

ular (White et al., 1992). It should be noted, however, that an equivalently partitioned DGGs could be constructed using the icosahedron itself. The other regular polyhedra remain unexplored for DGGs construction, so we limit our discussion here to the platonic solids.

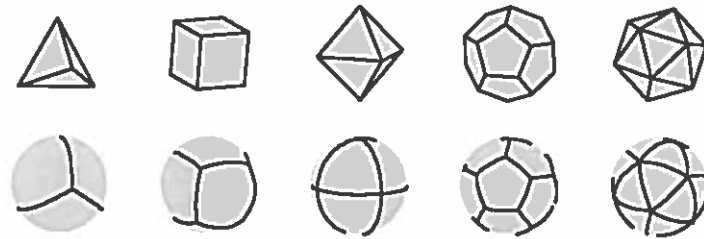


Figure 5. Planar and spherical versions of the five platonic solids: the tetrahedron, hexahedron (cube), octahedron, dodecahedron, and icosahedron.

In general, platonic solids with smaller faces reduce the distortion introduced when transforming between a face of the polyhedron and the corresponding spherical surface (White, Kimerling, Sahr, and Song, 1998). The tetrahedron and cube have the largest face size and are thus relatively poor base approximations for the sphere. But because the faces of the cube can be easily subdivided into square quadtrees, it was chosen as the base platonic solid by Alborzi and Samet (2000). The icosahedron has the smallest face size and, therefore, any DGGs defined on it tend to display relatively small distortions. The icosahedron is thus the most common choice for a base platonic solid. Geodesic DGGs based on the icosahedron include those of Williamson (1968), Sadournay et al. (1968), Baumgardner and Frederickson (1985), Sahr and White (1998), White et al. (1998), Fekete and Treinish (1990), Thuburn (1997), White (2000), Song et al. (2002), and (with an adjustment as discussed in the next section) Heikes and Randall (1995a, 1995b).

Dutton chose the octahedron as the base polyhedron for the Global Elevation Model (1984) and for the Quaternary Triangular Mesh (QTM) system (1999), while Goodchild and Yang (1992) based a similar system on it, and White (2000) used it as an

alternative base solid. The octahedron has the advantage that it can be oriented with vertices at the north and south poles, and at the intersection of the prime meridian and the equator, aligning its eight faces with the spherical octants formed by the equator and prime meridian. Given a point in geographic coordinates, it is then trivial to determine on which octahedron face the point lies, but, because the octahedron has larger faces than the icosahedron, projections defined on the faces of the octahedron tend to have higher distortion (White et al., 1998).

Wickman, Elvers, and Edvarson (1974) observe that if a point is placed in the center of each of the faces of a dodecahedron and then raised perpendicularly out to the surface of a circumscribed sphere ("stellated"), each of the 12 pentagonal faces becomes 5 isosceles triangles. The stellated dodecahedron thus has 60 triangular faces compared to the 20 faces of the icosahedron, and an equal area projection can be defined on the smaller faces of the stellated dodecahedron with lower distortion than on the icosahedron (e.g., Snyder, 1992). However, the triangular faces are no longer equilateral and therefore such a projection displays inconsistencies along the edges between faces.

Polyhedron Orientation

Once a base polyhedron is chosen, a fixed orientation relative to the actual surface of the Earth must be specified. Alborzi and Samet (2000) oriented the cube by placing face centers at the north and south poles. White et al. (1992) oriented the truncated icosahedron such that a hexagonal face covered the continental United States. Dutton (1984, 1999), and Goodchild and Yang (1992) oriented the octahedron so that its faces align with the octants formed by the equator and prime meridian. Wickman et al. (1974) oriented the dodecahedron by placing the center of a face at the north pole and a vertex of that face on the prime meridian, thus aligning with the prime meridian an edge of one of the triangles created by stellating the dodecahedron. In the case of the icosahedron, the most common orientation (Figure 6a) is to place a vertex at each of the poles and then align one of the edges emanating from the vertex at the north pole with the

prime meridian. This orientation is used by Williamson (1968), Sadournay et al. (1968), Fekete and Treinish (1990), and Thuburn (1997).

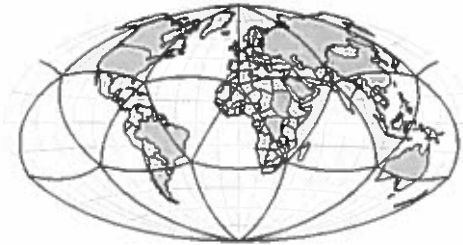


Figure 6a. Spherical icosahedron orientation with vertices at poles and an edge aligned with the prime meridian. Note the lack of symmetry about the equator.



Figure 6b. Spherical icosahedron oriented using Fuller's Dymaxion orientation. Note that all vertices fall in the ocean.

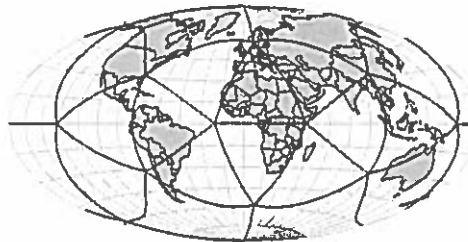


Figure 6c. Spherical icosahedron oriented for symmetry about equator by placing poles at edge midpoints. Note the symmetry about the equator and the single vertex falling on land.

Heikes and Randall's (1995a, b) icosahedron-based system was developed specifically for performing global climate change simulations. They note that in the common vertices-at-poles icosahedron placement (Figure 6a) the icosahedron is not symmetrical

about the equator. When a simulation on a DGGS with this orientation is initialized to a state symmetrical about the equator, and then allowed to run, it evolves into a state that is asymmetrical about the equator, presumably due to the asymmetry in the underlying icosahedron. To counter this effect they rotate the southern hemisphere of the icosahedron by 36 degrees, and the resulting “twisted icosahedron” is symmetrical about the equator.

Fuller (1975) chose an icosahedron orientation (Figure 6b) for his Dymaxion icosahedral map projection that places all 12 of the icosahedron vertices in the ocean so that the icosahedron can be unfolded onto the plane without ruptures in any landmass. This is the only known icosahedron orientation with this property. Note that one compact way of specifying the orientation of a platonic solid is by giving the geographic coordinates of one of the polyhedron’s vertices and the azimuth from that vertex to an adjacent vertex. For platonic solids this information will completely specify the position of all the other vertices. Using this form of specification, Fuller’s Dymaxion orientation can be constructed by placing one vertex at 5.2454° W longitude, 2.3009° N latitude and an adjacent one at an azimuth of 7.46658° from the first vertex.

We note that if the icosahedron is oriented so that the north and south poles lie on the midpoints of edges rather than at vertices, then it is symmetrical about the equator without further adjustment. While maintaining this property we can minimize the number of icosahedron vertices that fall on land, following Fuller’s lead. The minimal case appears to be an orientation (Figure 6c) that has only one vertex on land, in China’s Sichuan Province. This orientation can be constructed by placing one vertex at 11.25° E longitude, 58.28252° N latitude and an adjacent one at an azimuth of 0.0° from the first vertex.

Spatial Partitioning Method

Once we have a base regular polyhedron, we must next choose a method of subdividing this polyhedron to create multiple resolution discrete grids. In the case of platonic

solids one can define the subdivision methodology on a single face of the polyhedron or on a set of faces that constitute a unit that tiles the polyhedron, provided that the subdivision is symmetrical with respect to the face or tiling unit. Four partition topologies have been used: squares, triangles, diamonds, and hexagons.

Alborzi and Samet (2000) performed an aperture 4 subdivision to create a traditional square quadtree on each of the square faces of the cube. We have observed that the preferred choices for base platonic solids are the icosahedron, the octahedron, and the stellated dodecahedron, each of which has a triangular face. The obvious choice for a triangle is to subdivide it into smaller triangles. Like the square, an equilateral triangle can be divided into n^2 (for any positive integer n) smaller equilateral triangles by breaking each edge into n pieces and connecting the break points with lines parallel to the triangle edges (Figure 7). In geodesic dome literature this is referred to as a *Class I* or *alternate* subdivision (Kenner, 1976). Recursively subdividing the triangles thus obtained generates a congruent and aligned DGGs with aperture n^2 .

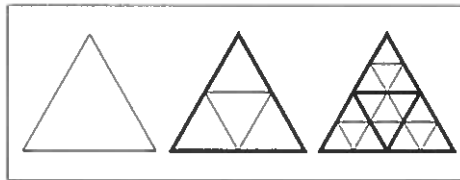


Figure 7. Three levels of a Class I aperture 4 triangle hierarchy defined on a single triangle face.

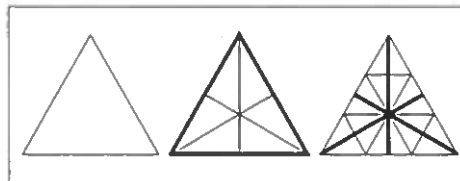


Figure 8. Three levels of a Class II aperture 3 triangle hierarchy defined on a single triangle face.

Small apertures have the advantage of generating more grid resolutions, thus giving applications more resolutions from which to choose. For congruent triangle

subdivision the smallest possible aperture is 4 ($n = 2$). This aperture also conveniently parallels the fourfold recursive subdivision of the square grid quadtree; many of the algorithms developed on the square grid quadtree are transferable to the triangle grid quadtree with only minor modifications (Fekete & Treinish, 1990; Dutton, 1999). This subdivision approach (Figure 7) was used by Wickman et al. (1974), Baumgardner and Frederickson (1985), Goodchild and Yang (1992), Dutton (1999), Fekete and Treinish (1990), White et al. (1998), and Song, Kimerling, and Sahr (2002). Congruent and unaligned Class I aperture 9 ($n = 3$) triangle hierarchies have been proposed by White et al. (1998) and Song et al. (2002).

An aperture 3 triangle subdivision is also possible. In this approach, referred to as the *Class II* or *triacon* subdivision (Kenner, 1976), each triangle edge is broken into $n = 2^m$ pieces (where m is a positive integer). Lines are then drawn perpendicular to the triangle edges to form the new triangle grid (Figure 8). The Class II breakdown is incongruent and unaligned. No Geodesic DGGs have been proposed based on this partition, though the vertices of a Class II breakdown have been used as cell points by Dutton (1984) and by Williamson (1968) to construct a dual hexagon grid.

Triangles have a number of disadvantages as the basis for a DGG. First, they are not squares; they are thus a foreign alternative for many potential users, and they do not display as efficiently as squares on common output display devices that are based on square lattices of pixels. Like square grids, they do not exhibit uniform adjacency, each cell having three edge and nine vertex neighbors. Unlike squares, the cells of triangle-based discrete grids do not have uniform orientation; as can be seen in Figures 7 and 8, some triangles point up while others point down, and many algorithms defined on triangle grids must take into account triangle orientation.

While the square is the most popular cell region shape for planar discrete grids, its geometry makes it unusable on the triangle-faced regular polyhedra that we have seen are preferred for constructing Geodesic DGGs. However, White (2000) notes that pairs of adjacent triangle faces may be combined to form a diamond or rhombus, and this diamond may be recursively sub-divided in a fashion analogous to the square quadtree

subdivision (Figure 9). When one begins with either the octahedron or icosahedron, this yields a congruent, unaligned Geodesic DGGs with aperture 4. Because diamond-based grids have a topology identical to square-based quadtree grids they can take direct advantage of the wealth of quadtree-based algorithms. But like square grids, they do not display uniform adjacency.

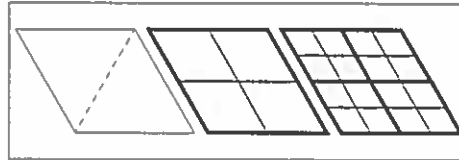


Figure 9. Three levels of an aperture 4 diamond hierarchy. The coarsest diamond resolution consists of two triangle faces as indicated.

As discussed in the last chapter, the hexagon exhibits a number of advantages as a planar tiling unit relative to squares and triangles. Studies by GIS researchers (Kimerling et al., 1999) and mathematicians (Saff & Kuijlaars, 1997) indicate that many of the advantages of planar hexagon grids may carry over into hexagon-based Geodesic DGGs. A hexagon-based grid has been adopted by the U.S. EPA for global sampling problems (White et al., 1992). And hexagon-based Geodesic DGGs have been proposed at least four times in the atmospheric modeling literature (Williamson, 1968; Sadournay et al., 1968; Heikes & Randall, 1995a, 1995b; Thuburn, 1997)—to our knowledge more often than any other Geodesic DGGs topology. It should be noted that it is impossible to completely tile a sphere with hexagons. When a base polyhedron is tiled with hexagon-subdivided triangle faces, a non-hexagon polygon will be formed at each of the polyhedron's vertices. The number of such polygons, corresponding to the number of polyhedron vertices, will remain constant regardless of grid resolution. In the case of an octahedron these polygons will be eight squares, in the case of the icosahedron they will be 12 pentagons.

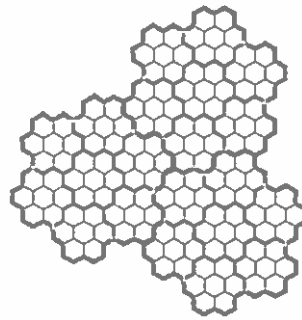


Figure 10. Seven-fold hexagon aggregation into coarser pseudo-hexagons.

While single-resolution, hexagon-based discrete grids are becoming increasingly popular, the use of multi-resolution, hexagon-based discrete grid systems has been hampered by the fact that congruent discrete grid systems cannot be built using hexagons; it is impossible to exactly decompose a hexagon into smaller hexagons (or, conversely, to aggregate small hexagons to form a larger one). Hexagons can be aggregated in groups of seven to form coarser resolution objects which are almost hexagons (Figure 10), and these can again be aggregated into pseudohexagons of even coarser resolution, and so on. Known as Generalized Balanced Ternary (Gibson & Lucas, 1982), this structure has become the most widely used planar multi-resolution, hexagon-based grid system. However, it has several problems as a general-purpose basis for spatial data structures. The first is that the cells are hexagons only at the finest resolution. Secondly, the finest resolution grid must be determined prior to creating the system, and once determined it is impossible to extend the system to finer resolution grids. Thirdly, the orientation of the tessellation rotates by about 19 degrees at each level of resolution. Finally, it does not appear to be possible to symmetrically tile triangular faces with such a hierarchy, which makes it unusable as a subdivision choice for a Geodesic DGGS.

There are, however, an infinite series of apertures that produce regular hierarchies of incongruent, aligned hexagon discrete grids; Dacey (1965) notes that aligned hexagon grids can be formed for any aperture $h^2 + hk + k^2$, where h and k are any positive integers. It should be noted that, as these hierarchies are incongruent, they do not

naturally induce hierarchical data structures which are trees, and thus common tree-based algorithms cannot be directly adapted for use on these hexagon hierarchies. But it should also be noted that, as indicated previously, traditional multi-resolution vector data structures such as the geographic coordinate system are also incongruent and aligned. This may indicate that hexagon grids are more appropriate for vector applications than congruent, unaligned triangle and diamond hierarchies.

Aperture 4 is the most common choice for hexagon-based DGGs. Figures 11 and 12 illustrate aperture 4 hexagon subdivisions corresponding to the Class I and Class II symmetry axes, respectively. The DGGs of Heikes and Randall (1995a) and Thurnham (1997) are Class I aperture 4 hexagon grids, while Williamson (1968) uses a Class II aperture 4 hexagon grid.

As noted above, small apertures have the advantage of allowing more potential grid choices. Aperture 3 is the smallest aperture that yields an aligned hexagon hierarchy (Figure 13). In aperture 3 hierarchies the orientation of hexagon grids alternates in successive resolutions between Class I and Class II. Aperture 3 hexagon Geodesic DGGs have been proposed by a number of researchers, including Sahr and White (1998).

White et al. (1992) proposed hexagon grids of aperture 3, 4, or 7, and White et al. (1998) discussed hexagon grids of aperture 4 (Class I) and 9 (Class II). Sadournay et al. (1968) used a Class I hexagon grid of arbitrary aperture, which is incongruent and unaligned. We refer to this approach as an *n-frequency* hierarchy. Note that it is possible to construct incongruent, unaligned *n-frequency* hierarchies using triangles and diamonds as well, though, to our knowledge, this has not been proposed.

Figure 14 illustrates the most common partitioning methods defined on an icosahedron and projected to the sphere using the inverse Icosahedral Snyder Equal Area (ISEA) Projection (Snyder, 1992).

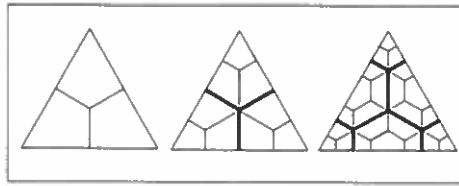


Figure 11. Three levels of a Class I aperture 4 hexagon hierarchy defined on a single triangle face.

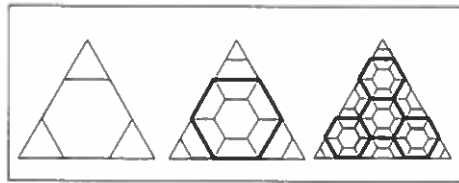


Figure 12. Three levels of a Class II aperture 4 hexagon hierarchy defined on a single triangle face.

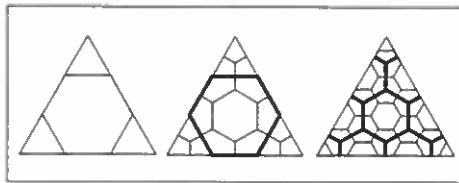


Figure 13. Three levels of an aperture 3 hexagon hierarchy defined on a single triangle face. Note the alternation of hexagon orientation (Class II, Class I, Class II, etc.) with successive resolutions.

Transformation

Once a partitioning method has been specified on a face or faces of the base polyhedron, a transformation must be chosen for creating a similar topology on the corresponding spherical or ellipsoidal surface. There are two basic types of approaches (Kimerling et al., 1999). *Direct spherical subdivision* approaches involve creating a partition directly on the spherical/ellipsoidal surface that maps to the corresponding partition on the planar face(s). *Map projection* approaches use an inverse map projection to trans-

form a partition defined on the planar face(s) to the sphere/ellipsoid. White et al. (1998) provide a comparison of the area and shape distortion that occurs under a number of different transformation choices.

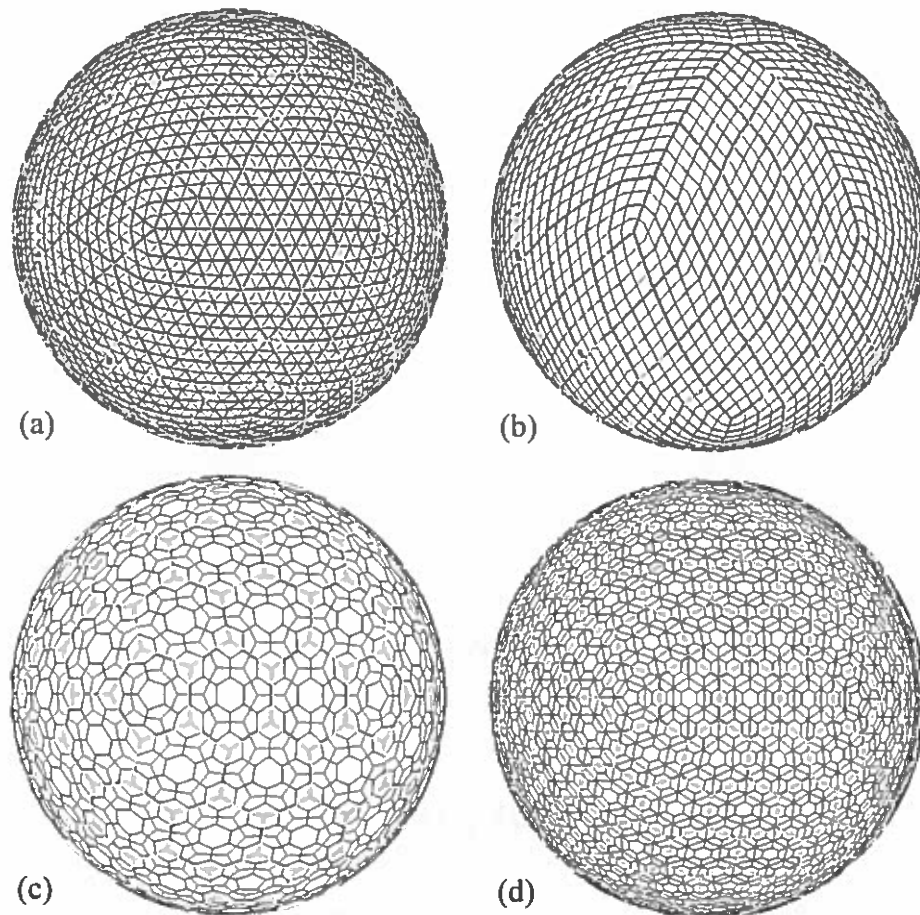


Figure 14. Three resolutions of icosahedron-based Geodesic DGGS's using four partition methods: (a) Class I aperture 4 triangle, (b) aperture 4 diamond, (c) aperture 3 hexagon, and (d) Class I aperture 4 hexagon.

Perhaps the simplest approach is to perform the desired partition directly on the spherical surface, using great circle arcs corresponding to the cell edges on the planar face(s). The aperture 4 Class I triangle subdivision can be performed on the sphere by connecting the midpoints of the edges of the base spherical triangle and then, recursively performing the same operation on each of the resulting triangles. This technique was used

by Baumgardner and Frederickson (1985) and Fekete and Treinish (1990). Dutton (1984) performed a Class II triangle subdivision on the surface of the octahedron and then adjusted the vertices to reflect the point elevations.

While this straightforward approach works for creating an aperture 4 triangle subdivision, it is important to note that, in general, sets of great circle arcs corresponding to the edges of planar triangle partitions do not intersect in points on the surface of the sphere, as they do on the plane. More complicated methods are needed to form spherical partitions analogous to some of the other planar partitions we have discussed.

Williamson (1968) used great circle arcs corresponding to two of the three sets of Class II triangle subdivision grid lines to determine a set of triangle vertices and then formed the last set of grid lines by connecting the existing vertices with great circle arcs. These triangle vertices form the center points of the dual Class II aperture 4 hexagon grid (the cell edges of which are not explicitly defined).

Sadournay et al. (1968) created an aperture m (where $m = n^2$ for some positive integer n) Class I triangle subdivision on the sphere by breaking each edge of the base spherical triangle into n segments and connecting the breakpoints of two of the edges with great circle arcs. These arcs are then subdivided evenly into segments corresponding to the planar subdivision. The resulting breakpoints form the centers of the dual Class I hexagon grid. Thuburn (1997) performed a Class I aperture 4 triangle subdivision and then calculated the spherical Voronoi cells of the triangle vertices to define the dual Class I aperture 4 hexagon grid.

A number of researchers have attempted to adjust the grids created using great circle arcs to meet application-specific criteria. For instance, for many applications it would be desirable for the cell regions of each discrete grid resolution to be equal in area; the grids discussed above do not have this property. Wickman et al. (1974) began by connecting the midpoints of the base spherical triangle to form the first resolution of a Class I aperture 4 triangle grid. They then broke each of the new edges at the midpoint into two great circle arcs and adjusted the position of the breakpoint to achieve equal area quasitriangles. This procedure is then applied recursively to yield an equal-area DGGs. Rather

than using great circle arcs for triangle subdivision, Song et al. (2002) proposed using small circle arcs optimized to achieve equal cell region areas.

Heikes and Randall (1995a) constructed a Class I aperture 4 hexagon grid by taking the spherical Voronoi of the vertices of a Class I aperture 4 triangle subdivision on their twisted icosahedron. They then adjusted the grid using an optimization scheme to improve its finite difference properties for use in global climate modeling.

White et al. (1998) evaluated a number of methods for constructing triangle subdivisions on spherical triangles and observed that using appropriate inverse map projections to transform a subdivided planar triangle onto a spherical triangle may be more efficient than using recursively defined procedures. Any projection may be used, provided that it maps the straight-line planar face edges to the great-circle arc edges of the corresponding spherical face.

There are at least four projections with this property. The common gnomonic projection has this property for all polyhedra but exhibits relatively large area and shape distortion. Snyder (1992) developed equal area projections defined on all of the platonic solids, but with greater shape distortion and more irregular spherical cell edges than the equal-area method of Song et al. (2002). On the icosahedron, the implementation of Fuller's Dymaxion map projection (Fuller, 1975) given in Gray (1995) also has the required property but with less area and shape distortion than the gnomonic projection and less shape distortion than Snyder's icosahedral projection, though the Fuller/Gray projection is not equal area. Goodchild and Yang (1992) used a Plate Carree projection to project the faces of the octahedron to the sphere, and Dutton (1999) developed the Zenithal OrthoTriangular (ZOT) projection for the same purpose.

White et al. (1998) constructed Class I aperture 4 and 9 triangle grids on planar icosahedral faces. They also constructed a Class I aperture 4 hexagon grid by taking the dual of the aperture 4 triangle grid and a Class II aperture 9 hexagon grid by aggregating the cells of the aperture 9 triangle grid. In all cases they transformed the resulting cells to the sphere, using direct spherical subdivision or the inverse gnomonic, Fuller/Gray, or Icosahedral Snyder Equal Area (ISEA) map projections.

White et al. (1992) and Alborzi and Samet (2000) used the inverse Lambert Azimuthal Equal Area projection to project the faces of the truncated icosahedron and cube to the sphere. White et al. (1992) noted, however, that this projection does not map the straight-line planar face edges to the corresponding great-circle arc edges and, therefore, does not create a true Geodesic DGGs.

Assigning Points to Grid Cells

When specified, the points associated with grid cells are usually chosen to be the center points of the cell regions. If an inverse map projection approach is used to transform the cells from the planar faces to the sphere, then it is often convenient to choose the center points of the planar cell regions (which do not, in general, correspond to the cell region centroids on the Earth's surface) so that the points form a regular lattice, at least on patches of the plane. If the cells are formed by direct spherical subdivision, the choice of points may be complicated by the counter-intuitive behavior of great-circle arcs described above. Gregory (1999) discussed several alternatives for point selection in the case of direct spherical subdivision. Dutton's (1984) GEM DGGs used points that are the vertices of a Class II triangle subdivision. As described in the previous sub-section, the hexagonal DGGs of Williamson (1968), Sadournay et al. (1968), Heikes and Randall (1995a), and Thuburn (1997) all specify cell center points as the vertices of a dual spherical triangle grid. The hexagonal grid cell boundaries, when specified, are created by calculating the associated spherical Voronoi cells.

Summary and Conclusions

Table 3 summarizes the design choices that define each of the Geodesic DGGs we have discussed. Note that the number of options employed to construct a Geodesic DGG is actually rather small, and certain choices clearly predominate in the existing designs. In particular, the icosahedron is clearly the popular choice for a base polyhe-

dron; it is used in 10 of the 16 listed grid designs. Methods based on direct spherical subdivision are employed by about half of the grid designs. Also popular are equal area transformations, which are used by six of the grids. The grid designs are almost evenly split between triangle and hexagon partitions, but the diamond partition is a recent design that may yet prove popular due to its direct relationship to the square quadtree.

We have shown that a Geodesic DGGS can be specified through a very small number of design choices, each of which is relatively independent of the others. In effect, future Geodesic DGGS designers may pick-and-choose from the menu of design choices to construct a DGGS to meet their specific application needs. As an example, let us take each of the design decisions in turn and attempt to construct a good general-purpose Geodesic DGGS.

First, due to its lower distortion characteristics we choose the icosahedron for our base platonic solid. We orient it with the north and south poles lying on edge midpoints, such that the resulting DGGS will be symmetrical about the equator. Next we select a suitable partition. The hexagon partition has numerous advantages, and we choose aperture 3, the smallest possible aligned hexagon aperture. Because equal-area cells are advantageous for many applications, we choose the inverse ISEA projection to transform the hexagon grid to the sphere, and we specify that each DGGS point lies at the center of the corresponding planar cell region. We call the resulting grid the ISEA Aperture 3 Hexagonal (ISEA3H) DGGS. Figure 15 shows the ETOPO5 global elevation data set (Hastings & Dunbar, 1998) binned into four resolutions of the ISEA3H DGGS. The elevation value for each ISEA3H cell was calculated by taking the arithmetic mean of all ETOPO5 data points that fall into that cell region.

Table 3. Summary of Geodesic DGGs design choices.

Reference	Base Polyhedron	Orientation	Partition	Transformation	Point Assignment
Alborzi & Samet (2000)	Cube	Poles on face centers	Aperture 4 Square	Lambert Azimuthal Equal Area	Not Specified
Baumgardner & Frederickson (1985)	Icosahedron	Unspecified	Aperture 4 Triangle (Class I)	Direct Spherical Subdivision	Not Specified
Dutton (1984)	Octahedron	Octant aligned	Not Specified (Implied Aperture 3 Hexagon)	Direct Spherical Subdivision	Class II Triangle Vertices
Dutton (1999)	Octahedron	Octant aligned	Aperture 4 Triangle (Class I)	ZOT	Not Specified
Fekete & Treinish (1990)	Icosahedron	Vertices at poles	Aperture 4 Triangle (Class I)	Direct Spherical Subdivision	Not Specified
Goodchild & Yang (1992)	Octahedron	Octant aligned	Aperture 4 Triangle (Class I)	Plate Carree	Not Specified
Heikes & Randall (1995a, 1995b)	Twisted Icosahedron	Vertices at poles	Twisted Aperture 4 Hexagon (Class I)	Optimized Direct Spherical Subdivision	Twisted Class I Aperture 4 Triangle Vertices

Table 3. Summary of Geodesic DGGs design choices (continued).

Reference	Base Polyhedron	Orientation	Partition	Transformation	Point Assignment
Sadourny et al. (1968)	Icosahedron	Vertices at poles	n -frequency Hexagon (Class I)	Direct Spherical Subdivision	n -frequency Class I Triangle Vertices
Sahr & White (1998)	Icosahedron	Equator symmetric	Aperture 3 Hexagon	ISEA	Cell Centers in ISEA Projection Space
Song et al. (2002)	Icosahedron	Unspecified	Aperture 4 or 9 Triangle (Class I)	Equal Area Small Circle Subdivision	Not Specified
Thuburn (1997)	Icosahedron	Vertices at poles	Aperture 4 Hexagon (Class I)	Direct Spherical Subdivision	Class I Aperture 4 Triangle Vertices
White et al. (1992)	Truncated Icosahedron	Hexagon face covering CONUS	Aperture 3, 4, or 7 Hexagon	Lambert Azimuthal Equal Area	Not Specified

Table 3. Summary of Geodesic DGGs design choices (continued).

Reference	Base Polyhedron	Orientation	Partition	Transformation	Point Assignment
White et al. (1998)	Icosahedron	Unspecified	Class I Aperture 4 or 9 Triangle or Class I Aperture 4 or Class II Aperture 9 Hexagon	Direct Spherical Subdivision, Gnomonic, ISEA, or Fuller/Gray	Not Specified
White (2000)	Icosahedron or Octahedron	Unspecified	Aperture 4 Diamond	Not Specified	Not Specified
Wickman et al. (1974)	Stellated Dodecahedron	Stellation vertex on pole	Aperture 4 Triangle (Class I)	Area-adjusted Direct Spherical Subdivision	Not Specified
Williamson (1968)	Icosahedron	Vertices or Face Centers at Poles	Not Specified (Implied Class II Aperture 4 Hexagon)	Direct Spherical Subdivision	Modified Class II Triangle Vertices

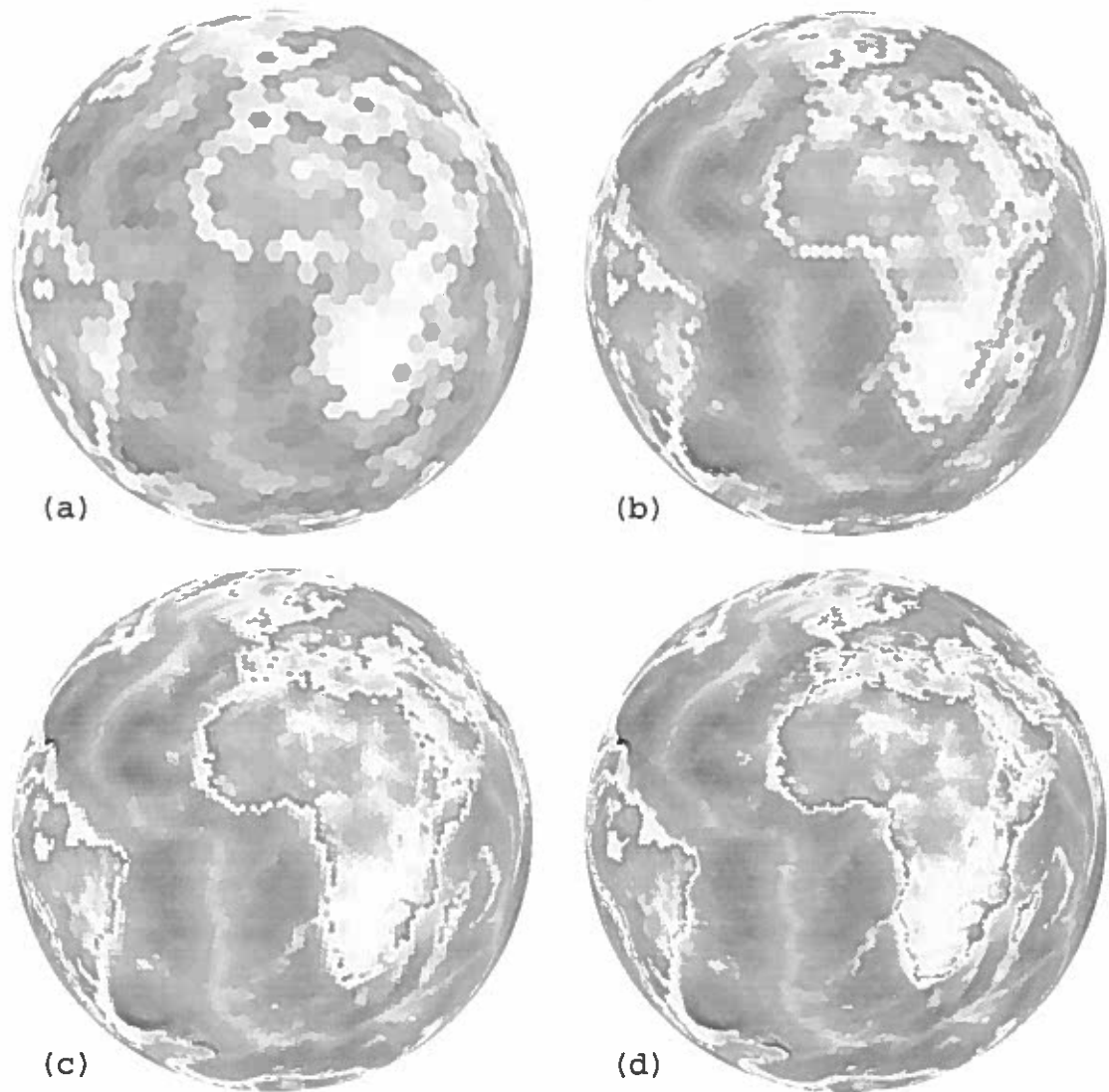


Figure 15. ETOPO5 5' global elevation data binned into the ISEA3H Geodesic DGGS at four resolutions with approximate hexagon areas of: (a) 210,000 km², (b) 70,000 km², (c) 23,000 km², and (d) 7,800 km².

Directions for Further Research

While such studies as White et al. (1998), Kimerling et al. (1999), Clarke (2002), and the current work have made significant steps in defining and evaluating existing

DGGS alternatives there remain a number of areas that we believe require further research. First, it should be noted that additional research may reveal new design choice alternatives that are superior to those already proposed. In particular, we feel that further research into transformations for Geodesic DGGS definition is required. For example, a DGGS projection that is equal area, but has less shape distortion than the ISEA projection, would be very desirable. Additionally, the grids discussed here are defined with reference to the sphere; many applications will require more accurate definitions referenced to ellipsoids. And as specific grids are chosen for practical use efficient transformations must be defined that will allow data to be moved between grids while preserving data quality.

Existing studies have treated DGGSs from the perspective of the broader GIS community, but effective evaluation of design alternatives can only take place in the context of specific applications and end-user communities. In particular, the computer data structures community has yet to play a significant role in DGGS evaluation. Input from this community, which should play a key role in making appropriate design choices in the future, has been primarily limited to the adaptation of quadtree algorithms to aperture 4 triangle grids (in particular the QTM DGGS of Dutton, 1999). Hexagon-based Geodesic DGGSs, which have clear advantages for many end-users, remain largely ignored. A significant effort must be made by the data structures community to develop and evaluate algorithms for the regular, but non-tree, hierarchies they form.

In order to conduct empirical evaluations of the relative performance of various DGGS topologies we will require an implementation of DGGS's designed specifically to facilitate experiments. In the next chapter we will describe just such an implementation.

CHAPTER III

Nulib: A Topology-Independent DGGS Architecture

Motivation

Previous DGGS researchers have primarily concentrated on implementing and analyzing a single DGGS topology, which has hampered efforts to compare and contrast the many varieties of DGGSs. The burgeoning interest in these data structures by end-users has increased the need for software architectures that would facilitate attempts to prototype, visualize, and analyze particular DGGS instances. Such an architecture should provide a common, consistent interface for constructing and interacting with DGGS, ideally in a manner that would allow client applications to build prototype applications and construct grid performance tests in a topology-independent way. Such an interface should include at least a core set of the functionality expected by common spatial data structure clients. At the same time, it should provide a consistent interface to non-DGGS geospatial data structures to facilitate access to existing capabilities that may be defined on those structures, for example data import or visualization. In this section we will give a specification for a software architecture that we believe meets these requirements. We have implemented the described specification as a C++ software library we call *nulib* (after *Nuit*, the Egyptian goddess of undifferentiated space). We will begin by discussing the design and implementation of the core *nulib* library. We will then demonstrate the use of the library in building one important class of DGGS application: a DGGS-based topology-independent discrete simulation engine.

Implementing the Nulib Core

Geospatial Data Structures ADT

In Chapter 1 we gave a formal definition of an ADT for structured geospatial data structures. Since we wish our library to allow us to potentially construct any structured geospatial data structure, this definition provides us with our minimal design requirements. In this section we will revisit the terminology we developed in Chapter 1, but specifically from the perspective of implementing our core library functionality.

The primary concept that any spatial data structure must capture is that of *location*. Location is independent of any particular representation of it. We can distinguish between a location and a particular computer representation of that location in the context of a particular reference frame (RF). We will use the term *address* to refer to a computer representation of a particular location equivalence class in a particular location RF. Such a representation would normally have no meaning outside the context of that particular frame of reference.

Ideally, domain-specific client applications interface with spatial data structures using locations. Internally each location consists of an address and a pointer to the RF under which that address is defined. If a sufficient set of operations are defined on locations, then applications should not need to access—or even have knowledge of the nature of—the current address value. For example, the RF (and thus potentially both the form and value of the address) under which the location is currently defined might change, but this should have no effect on client applications. Since locations cannot be defined independent of an RF, we can use RF objects as factories for creating location objects. RFs are implemented generically in nulib using a templated address type.

In order for such a location system to be generally useful it must define a reasonable set of functionality on locations. This functionality is then implemented internally in the appropriate RFs in terms of operations on RF addresses.

In order for a newly introduced RF to be useful it must define at least one quanti-

zation operator that converts locations from some other, previously known, RF into addresses in the new RF. Take any two reference frames $RF1$ and $RF2$ defined on the same spatial domain. Given a location consisting of an address defined on $RF1$, it should be possible to define a *converter* function that determines the value of the corresponding address in $RF2$. Let us suppose that converter functions are defined from $RF1$ to $RF2$, and from $RF2$ back to $RF1$. Assume also that converter functions are defined from/to $RF2$ to/from a third reference frame, $RF3$. Whenever a series of converters that yield a two-way transformation path between two reference frames exists we say that the frames are *connected*. Thus in this example $RF1$ and $RF2$ are connected, $RF2$ and $RF3$ are connected, and $RF1$ and $RF3$ are connected (via $RF2$). Given a network of connected RFs and an unconnected RF, we need only provide converters to/from the new RF from/to a connected RF and the new RF will be connected to all other RFs in the network. The connection path can then be followed automatically to perform location conversions without forcing the user to explicitly define and invoke intermediate converters.

Complex geospatial data structures such as DGGs may be constructed from a set of sub-RFs, with multiple conversion paths defined between these sub-RFs. In such systems it may not be a priori clear which sequence of conversions will be most efficient and involve the least quantification error. Maintaining an explicit network of existing converters gives us a convenient place to track the accumulation of quantification error, and to implement conversion tracing and performance monitoring. Once a particular conversion sequence is chosen the entire sequence can be implemented in a single converter, thus removing any inefficiency associated with multi-step conversion.

The explicit storage of an RF pointer in each location is wasteful in situations where only a single RF is under consideration. In such cases it makes sense to use a *location vector*, a vector of locations all defined on the same RF, which can be stored together as a vector of addresses with a single RF pointer. Location vectors can be converted by applying the appropriate converter to each of the individual addresses. Location vectors can be used to specify a vector of point locations, or they can specify a polyline, or the boundary of a polygon.

In earlier versions of our architecture we made a fundamental distinction between explicitly discrete RFs, such as planar grids or DGGs, and RFs that attempt to mimic real number coordinate systems, e.g., vectors of floating point numbers. A further fundamental distinction was made between single resolution RFs and multi-resolution RFs. These distinctions led to an unnecessarily complex class hierarchy. We have since realized that any computer-based RF can be viewed as discrete—since any address representation on a computer must by definition be discrete and finite—and multi-resolution—even if only a single resolution of the RF actually exists.

If we assume that every RF is discrete, then each location corresponds to a region on the spatial domain of interest. This region must itself be defined in the context of some RF, which may or may not be the same RF as the original location specification. Thus for all RFs we specify a back-frame, which is the default frame for that RF for the definition of location regions. In addition, since it is often useful to have a point representation of any location, we designate the back-frame as the default frame for such a definition. Thus a location is a functional match for a DGGs *cell* (as defined in Chapter 2). This terminology is reflected in the nolib implementation, and we will use it for the remainder of our discussion.

The minimal functional interface for an RF consists of the following operations on cells (regions and sets are specified as location vectors):

1. quantization (conversion).
2. region format specification (in back-frame RF).
3. point format specification (in back-frame RF).
4. metric distance.

As discussed in Chapter 1, the availability of a metric distance operator for an RF allows nolib to define default equality and adjacency operators for that RF.

For any RF we can define one or more *sequence RFs* that have a linear address form. These sequence RFs have as their back-frame the desired primary RF and specify a total order that is bounded above and below on that RF. Sequence RFs specify address increment and decrement methods that iterate over all legal addresses in the primary RF.

Hierarchical Multi-Resolution RF Operators

Traditional multi-resolution structured geospatial data structures, such as the quadtree, form hierarchies based on spatial containment that are trees. That is, given any resolution k cell, there is one and only one resolution $k-1$ cell whose region entirely contains the region of that cell. The coarser and finer resolution cells have a strict parent/child relationship that allows us to define a tree structure. As we have seen in Chapter 2, however, the hierarchies implied by multi-resolution hexagon grids do not define traditional tree structures. The region associated with a resolution k hexagon cell may or may not be entirely contained in a single resolution $k-1$ cell region.

In order to be able to define the notion of hierarchical operators in a topology-independent manner we must extend our notion of hierarchy to include non-tree structures. We call this more general definition a *spatial hierarchy*. A spatial hierarchy is defined on a DGGS by determining, for each resolution k cell C in a DGGS, the following four sets:

1. *Parents*. The set of all resolution $k-1$ cells whose interiors intersect the interior of C .
2. *Neighbors*. The set of all resolution k cells that are neighbors of C , defined using resolution k grid adjacency operator.
3. *Interior Children*. The set of all resolution $k+1$ cells whose interiors are entirely contained in the interior of C .
4. *Boundary Children*. The set of all resolution $k+1$ cells whose interiors intersect, but are not entirely contained in, the interior of C .

We further define the *hierarchical adjacency set* for the cell C to be the union of these four sets.

Additionally, the natural concept of ancestors and descendents can be used to extend the concept of hierarchical neighborhood beyond the simple adjacency case, though we will not do so here.

As noted above, we can treat all RFs as multi-resolution RFs; we thus add the

determination of spatial hierarchy sets to the minimal RF interface defined in the last section. Note that in the case of single-resolution RFs all sets except the set of neighbors will be empty for all cells.

Nulib implements the spatial hierarchy operators for the primary icosahedral DGGS topologies given in Chapter 2. These are aperture 4 triangle (as illustrated in Figures 7 and 14(a)), aperture 4 diamond (Figures 9 and 14(b)), aperture 3 hexagon (Figures 12 and 14(c)), and aperture 4 hexagon (Figures 11 and 14(d)). Figures 16 and 17 illustrate the hierarchical adjacency set for planar aperture 4 triangle and aperture 4 diamond topologies respectively. Note that for both of these topologies the set of boundary children is empty. On the plane both the diamond and triangle grids have two valid metric definitions, depending on whether or not we count cells that share only a vertex as neighbors. On the spherical icosahedron the topology of triangles or diamonds adjacent to the icosahedral vertices differs from those elsewhere on the icosahedron; these cells have an irregular number of vertex neighbors. For this reason we will restrict our attention to the edge neighbor metrics only.

Figures 18 and 19 illustrate the planar aperture 3 and 4 hexagon topologies respectively. Note that for the hexagon topologies the number of parent cells varies; in the aperture 4 case a cell may have either one or two parents, and in the aperture 3 case a cell may have one or three parents. Additionally, a hexagon grid can only be tiled onto the spherical icosahedron by making each of the twelve cells centered on the icosahedron vertices into pentagons. These cells will have one less neighbor and boundary children cells in their hierarchical adjacency sets than the hexagonal cells that tile the remainder of the spherical icosahedron.

Since the core nulib library is independent of any particular address form, the spatial hierarchies were determined using geometric means. Specific hierarchical addressing systems should override these default definitions with more efficient address-based calculations as required.

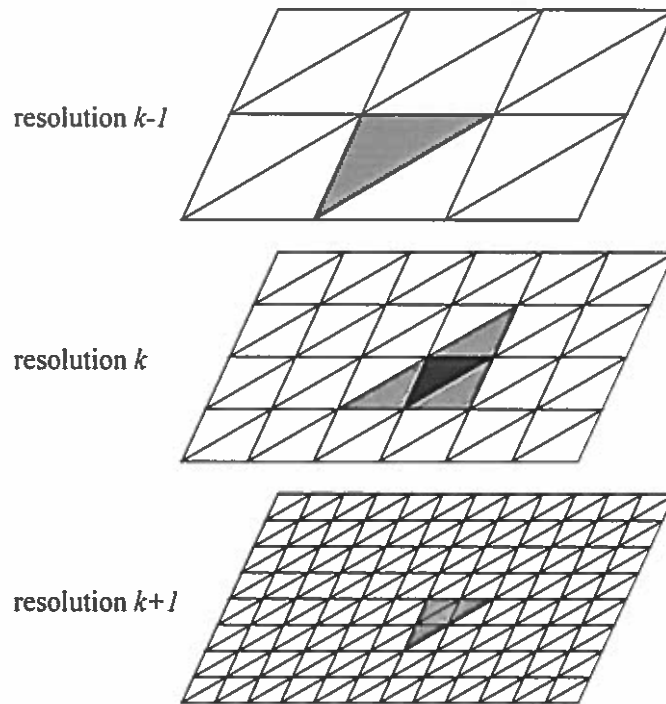


Figure 16. Hierarchical adjacency set for the planar aperture 4 triangle topology. The focal cell is given in black at resolution k . The parents, neighbors, and interior children sets are given in gray at resolutions $k-1$, k , and $k+1$ respectively. Note that the boundary children set is empty under this topology.

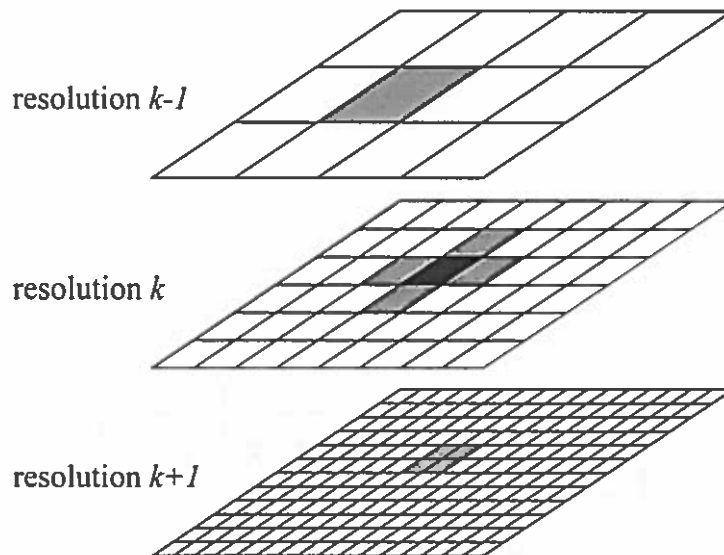


Figure 17. Hierarchical adjacency set for the planar aperture 4 diamond topology. The focal cell is given in black at resolution k . The parents, neighbors, and interior children sets are given in gray at resolutions $k-1$, k , and $k+1$ respectively. Note that the boundary children set is empty under this topology.

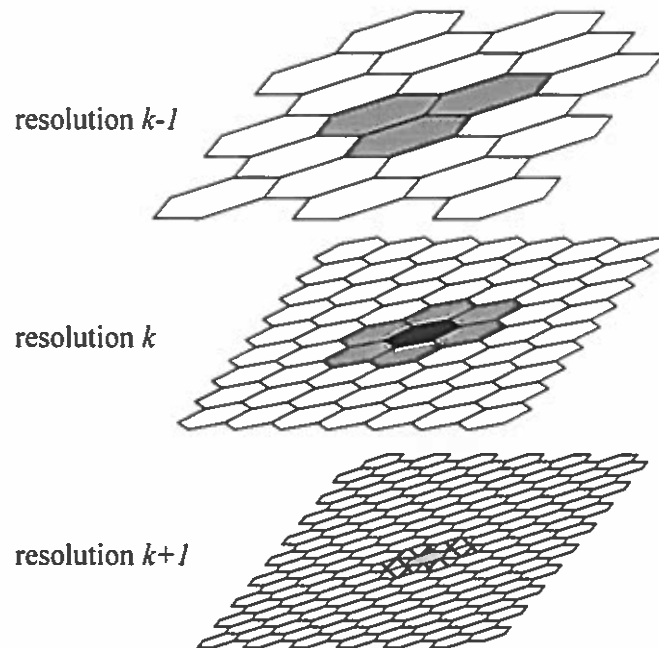


Figure 18. Hierarchical adjacency set for the planar aperture 3 hexagon topology. The focal cell is given in black at resolution k . The parents, neighbors, and interior children sets are given in gray at resolutions $k-1$, k , and $k+1$ respectively. The boundary children set is indicated by cross-hatching in resolution $k+1$.

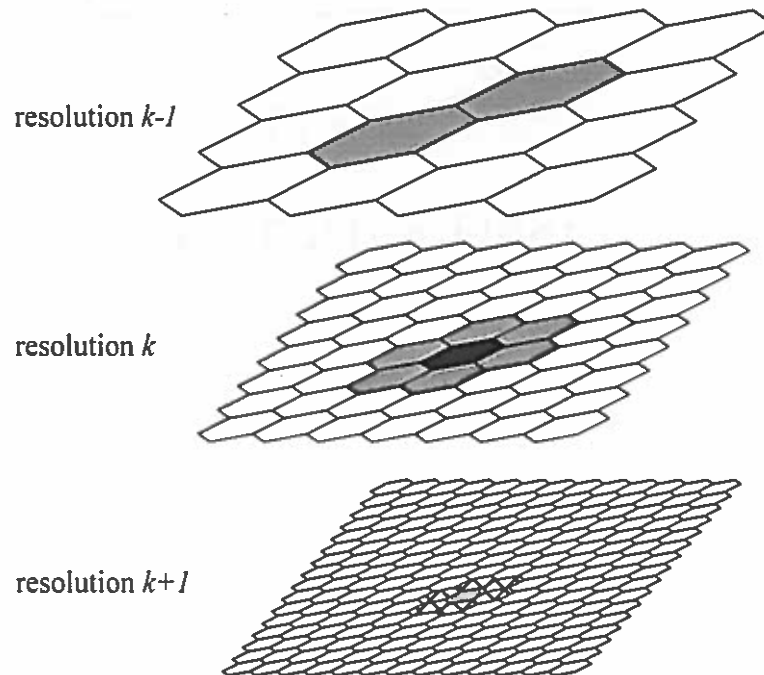


Figure 19. Hierarchical adjacency set for the planar aperture 4 hexagon topology. The focal cell is given in black at resolution k . The parents, neighbors, and interior children sets are given in gray at resolutions $k-1$, k , and $k+1$ respectively. The boundary children set is indicated by cross-hatching in resolution $k+1$.

Spatial Database Functionality

Up to this point we have treated RFs as mathematical entities, with no storage space associated with particular RF cell addresses. Such RFs are extremely useful as intermediate conversion spaces, as well as providing a means for performing calculations on conceptually infinite address spaces. But many applications, such as simulations or spatial databases, require an association between cells and potentially arbitrary data contents. For this we introduce the *database RF*. Objects of this class are associated with a particular RF and provide methods that, given a cell defined on that RF, will as desired set, get, replace, or delete the data contents of that cell. In nulib the database RF is implemented generically using a templated content type.

Applications of Nulib

The nulib implementation of this specification has been used to implement the four primary icosahedral DGGs topologies discussed in the last chapter, using the ISEA projection. In order to maximize code reuse in the construction of the grids all of the DGGs topologies are addressed using a quadrilateral two-dimensional integer system (see chapter 6).

The nulib architecture provides the necessary functionality to allow us to implement a discrete spatial simulation engine; at the same time the construction of such an engine provides a test of the expressive power and functionality of the nulib architecture.

We define a generic *simulation cell* that provides an abstract interface to the following state data and methods, which must be fully defined by a particular discrete simulation instance:

1. a set of static state data, the value of which does not change over time.
2. a set of variable state data that can change over time. Each cell maintains two sets of identically formed variable state data to support double-buffering (as described below).

3. a graphical cell representation and a method which sets the cell's graphical display parameters based on the cell's current state.
4. an *initialize* method, which must be called once before the first time a simulation involving this cell is executed.
5. a *reset* method, which must be called before each simulation run.
6. a *process* method, which is called on the cell at each time step. In a double-buffered simulation the cell uses the data from the current variable state buffers of itself and its neighbors to calculate its next variable state, which it stores in the second variable state buffer. After all cells have been processed this state data is then copied back into the current variable state in preparation for the next time step.
7. a *postprocess* method, which is called on the cell after a simulation run has completed execution.

A *simulation RF* is a database RF of simulation cells with the following additional functionality:

1. a discrete clock which allows the client application to specify a simulation start time, time increment size, and stop time.
2. an *initialize* method that initializes the clock, and initializes the grid cells by using the sequence iterator facility to call each cell's *initialize* method.
3. a *reset* method that re-initializes the clock and calls each cell's *reset* method.
4. a *process* method that has the pseudocode description:

```

time = initialTime
while time <= stopTime
    foreach cell: cell.process()
    foreach cell: cell.setGraphicState()
    grid.display()
    foreach cell: cell.swapbuffers()
    time = time + timeIncrement
endWhile

```

5. a *postprocess* method that calls each cell's postprocess method.

Each of the simulation engine methods may be redefined as desired by specific simulation engine instances.

Given the nulib implementation of DGGS topologies and this generic discrete simulation interface we are now in a position to implement any specific discrete simulation. If we specify the simulation cell processing using only interface functionality defined on nulib cells (as described above), we can then change the underlying spatial model of our simulation by simply switching the RF that is associated with the underlying database RF. No changes need to be made to the simulation specification itself. This allows us to easily experiment with alternative DGGSs. In the next chapter we will demonstrate this capability by using our nulib simulation architecture to implement a simple discrete simulation model and comparing the results of running this simulation on different DGGS topologies.

In a broader sense, we hope that our implementation of a topology independent DGGS software architecture will enable further studies comparing the use of various DGGS topologies for the wide range of applications for which they may be appropriate, and for which such studies are sorely lacking.

CHAPTER IV

Spatially Hierarchical Cellular Automata on Discrete Global Grids

Acknowledgements

The material in this chapter is being prepared for publication submission with my co-author, A. Ross Kiester. The research was conducted in partnership with Dr. Kiester, and he prepared figures 21 and 22. All other figures were created by the present author.

Introduction

One important application of DGGs is as a spatial substrate for the dynamic discrete simulation and analysis of global processes. DGGs allow for the definition of such simulations on a surface that is topologically equivalent to the spherical surface of the earth. Many proposed DGGs are multi-resolution, with regular relationships between grid cells at differing resolutions, and they thus hold the potential of providing a consistent spatial model for the simulation of multi-scale phenomenon, with hierarchical relationships between spatially coincident processes at different scales.

When constructing traditional single resolution simulations the choice of topology can have important effects. In particular, while the traditional square grid remains the primary spatial substrate for discrete simulations on the plane, grids based on hexagons have received a great deal of recent interest as a basis for simulations both on the plane and on the sphere. For example, we observed in Chapter 2 that hexagon-based DGGs have been proposed more often than any other DGG topology for atmospheric model-

ing simulations. The increased interest in hexagon grids appears to be primarily a consequence of the fact that cells in hexagon grids display uniform adjacency, as discussed in Chapter 1.

The use of DGGs as the basis for multi-scale discrete global simulations has not previously been attempted. Thus we know virtually nothing about the potential consequences that a particular choice of hierarchical DGG topology may have. In this chapter we will report the development and results of the first experiments designed to begin to give us some understanding of these issues. For these initial experiments we have chosen the simplest of discrete simulations: a cellular automata (CA). In previous research (Kiestler & Sahr, 2000) we developed a topology-independent multi-scale analog of traditional single-resolution CA. We begin here by extending that definition to the sphere by defining multi-scale CA for the primary DGG topologies. We then describe the development of a software architecture designed to facilitate the implementation of topology-independent discrete simulations on DGGs. This architecture is used to implement a specific CA, a spherical multi-scale topology-independent analog of Conway's Game of Life. Finally, we discuss the results of our first experiments running this simulation.

Extending Cellular Automata to DGGs

A CA (von Neumann, 1966) is a discrete dynamical mathematical structure defined by four properties:

1. *A State Space.* The set of all possible values for a single grid cell. The simplest and most commonly employed state space is binary, often indicated by saying that a cell is "alive" or "dead."
2. *A Grid Topology.* The most common two-dimensional grid topology is a grid of squares with either the D8 or D4 metric (as illustrated in Figure 1). Other grid topologies, such as the hexagon, have received limited attention.
3. *A Rule Set.* For each cell at each discrete time t , the rule set specifies the state of the cell at time step $t+1$. The new state is a function of the current states of

the cell and of the cells in a neighborhood of that cell. In the most common case only the states of the nearest neighbors of the cell (those at a metric distance of one) are considered.

4. *An Initial Condition.* At time $t = 0$ each cell is assigned an initial state from the state space.

By far the most widely studied CA is John Conway's Game of Life; this makes it a good candidate for our first set of experiments. As first described by Gardner (1970), Life is defined on a single resolution square grid with the D8 metric and a binary state (*alive* or *dead*). Each cell is initialized with a random value. At each time step, the cell determines its next state based on its current state and the number of its neighbors which are currently alive, using the state transition table given in Table 4.

Table 4. Life state transition table (with D8 metric)

state at time i	# of alive neighbors	state at time $i+1$
dead	3	alive
dead	1, 2, 4-8	dead
alive	2, 3	alive
alive	1, 4-8	dead

The state transition table given in Table 4 assumes that each cell has 8 neighbors. In order to implement Life in a topology-independent manner we must generalize this table so that it can be used with the variable neighbor counts generated by the alternate DGGs topologies. One simple way to do this is to change the second column so that it uses the fraction of adjacent cells currently alive, rather than a discrete count of alive neighbors. The resulting state transition diagram is given in Table 5.

Table 5. Generalized Life state transition table

state at time <i>i</i>	$x = \text{\#alive} / \text{\#total}$ (of adjacent cells)	state at time <i>i+1</i>
dead	$2/8 < x \leq 3/8$	alive
dead	$x \leq 2/8$ or $x > 3/8$	dead
alive	$1/8 < x \leq 3/8$	alive
alive	$x \leq 1/8$ or $x > 3/8$	dead

In order to extend the CA concept to multi-scale grids we expand the notion of adjacency used in specifying a rule set to include the concept of hierarchical adjacency sets as defined in the last chapter. Then the cell state at each time step is a function of the current states of the cell and of the cells in the hierarchical adjacency set of that cell, rather than just of the states of its neighbors at the same resolution. A rule set may treat all cells in the hierarchical adjacency set in an equivalent manner, or it may make a distinction by, for example, weighting cells in each of the sub-sets differently.

For our first experiments we chose to treat all hierarchical adjacency set members equally. This allows us to use Table 5 to define our state transitions.

We can now implement a DGGS-based, topology-independent, multi-scale version of Life using the nulib discrete simulation interface specified in the last chapter. Each Life grid cell has no static state data and variable state data consisting of a single boolean variable that indicates the current state (alive or dead) of the cell. In both the initialize and reset methods each cell is initialized to a random state based on a user-specified percentage of desired initial live cells. The process method implements the state transition diagram given in Table 5, using the topology-independent nulib interface to the underlying DGGS RF. We define no postprocess method for this cell type. Finally, we specify the graphic state of alive and dead cells to be solid black and solid white cell polygons respectively. Given this cell definition, no redefinition of the basic simulation engine functionality is needed.

Results

Experiments were run with the major DGGs topologies described in chapter 2: aperture 4 triangle, aperture 4 diamond, aperture 3 hexagon, and aperture 4 hexagon. After some initial exploratory runs we arbitrarily chose a maximum of 50,000 time steps for each run and four initializing percentages: 6%, 11%, 18%, and 25%. A four-resolution grid was run for all topologies. Additional runs were made with three- and five-resolution hexagon grids using 11% and 18% initialization rates. For each set of parameters 24 simulations were performed (each using a unique random number seed). A total of 28 separate parameter combinations were simulated for a total of 672 runs. Figure 20 shows a four-resolution aperture 3 hexagon grid initialized with an initialization percentage of 6%.

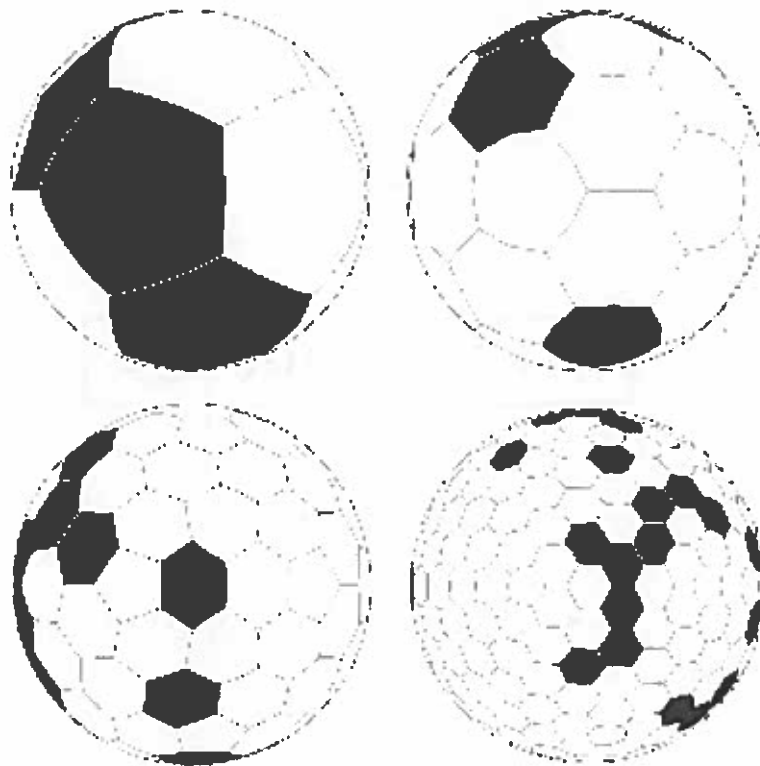


Figure 20. Four-resolution aperture 3 hexagon grid initialized with an initialization percentage of 6%. Live cells are shown in black.

Runs were continued until they reached the maximum time of 50,000 time steps or until they achieved a steady, unchanging state, whichever came first. Table 6 gives the mean number of time steps for all four-resolution runs. As illustrated, while the great majority of hexagon runs continued running until the maximum time step, every triangle and diamond run quickly achieved a steady state. Indeed, the longest-running triangle run terminated in 76 steps, while the longest running diamond run terminated in 33 steps. This result was not anticipated. Note that the hierarchical adjacency set of an aperture 4 triangle cell consists of 8 cells (1 parent, 3 neighbors, and 4 interior children), which matches the number of neighbors in the traditional Game of Life. From a purely numerical standpoint we might expect the triangle grid to perform similarly to traditional Life, as would the aperture 4 diamond topology with 9 adjacent cells (1 parent, 4 neighbors, and 4 interior children). Observation revealed that the cells at the finest resolutions were dying off immediately, with no cells coming alive, followed by a similar pattern in successively coarser resolutions. This behavior makes sense when we realize that the finest resolution cells have no children; thus a finest resolution triangle cell has only 4 adjacent cells (1 parent and 3 neighbors) and a finest resolution diamond cell has only 5 adjacent neighbors (1 parent and 4 neighbors). Given the ranges in Table 5 this makes it impossible for dead cells to become alive at the finest resolution, and unlikely that live cells will remain alive. Thus, while the spherical surface of a DGGS has no boundary effects within a given resolution, there is a definite boundary effect at the finest resolution of the multi-resolution spatial hierarchy.

Table 6. Mean number of time steps achieved by four-resolution runs by topology and initializing percentage.

Topology	Initializing Percentage			
	6%	11%	18%	25%
triangle	4	9	19	29
diamond	3	6	15	20
hexagon (3)	12,503	45,834	50,000	50,000
hexagon (4)	33,335	50,000	50,000	50,000

The number of cells at a given resolution that are alive (or dead) as a function of time in the simulations form a time series. Figure 21 shows a sample time series for the fourth (finest) resolution of a four-resolution aperture 3 hexagon case. This series was randomly initialized to 18% live. The simulation ran for 50,000 time steps. Notice that while the series is mostly constrained, occasional values higher and lower than usual occur. We do not know how long this series would have continued. We have run similar simulations for 250,000 time steps that were exhibiting similar dynamics throughout.

These time series may be analyzed using the methods of time series analysis (see, e.g. Shumway, 2000). The autocorrelation function (ACF) is, perhaps, the easiest way to look for temporal structure in data. It simply measures the strength of the statistical correlation between sets of pairs of points separated by a given lag or distance in time. Like the usual statistical (Pearson) correlation, the ACF varies between -1 and 1. Significant departures from 0 indicate temporal structure in the data. Significant ACF values beyond lag = 0 can be interpreted as a kind of memory in an otherwise apparently noisy process. In Figure 22 we give the ACF for some representative sample series for lags up to 200 from simulations with different topologies. Each topology was run with four resolutions and a random 18% of cells initialized to live. Data for the planar cases is from Kiester & Sahr, 2000. ACF values above or below the dashed blue lines are statistically significant. Note that both planar cases show more autocorrelation than the spherical cases, with a striking difference between planar hexagon aperture 3 and planar hexagon aperture 4. Both spherical cases show very little autocorrelation.

Hex Aperture 4 Resolution 4 of 4

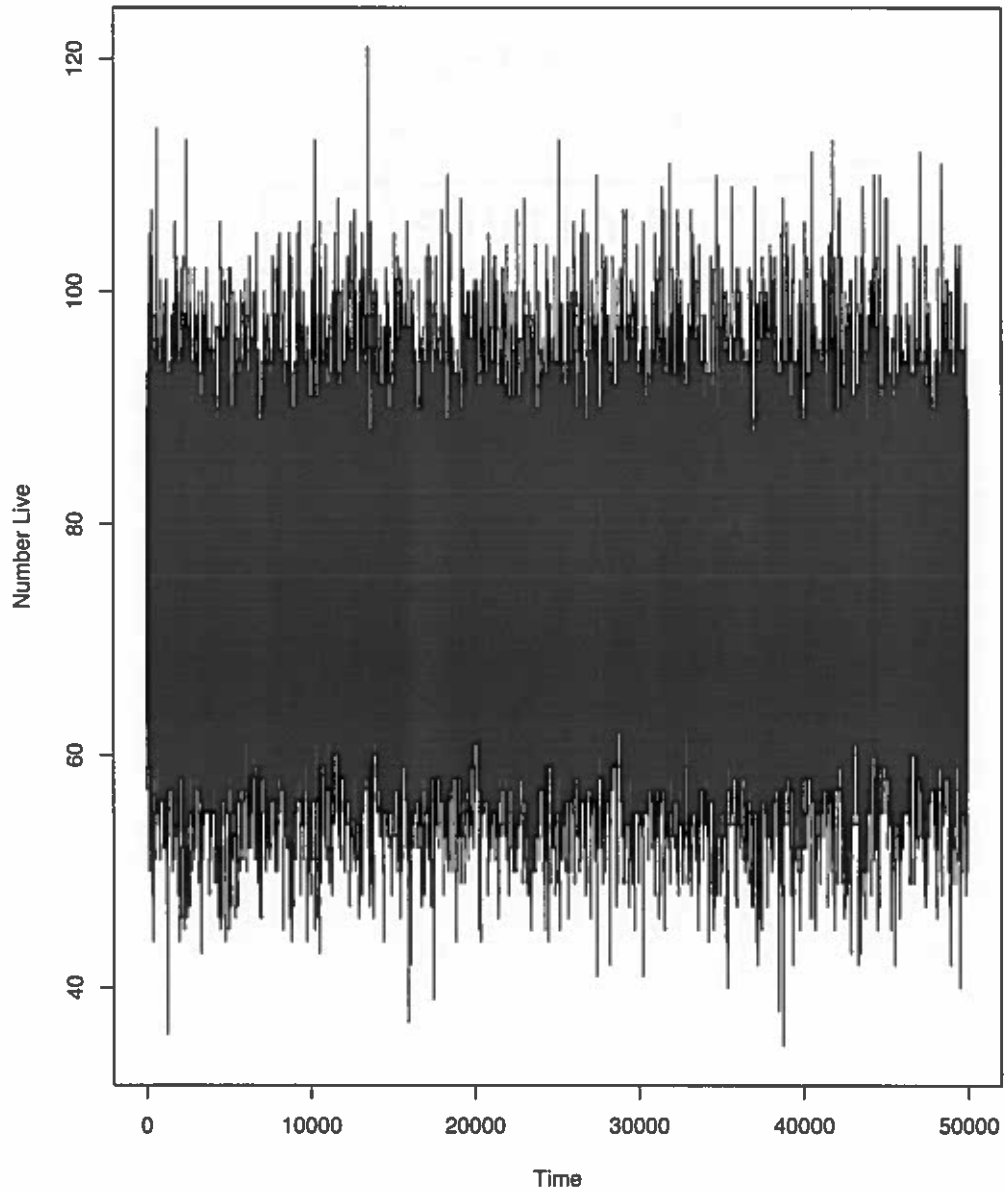


Figure 21. Sample time series for the fourth (finest) resolution of a four-resolution aperture 3 hexagon case.

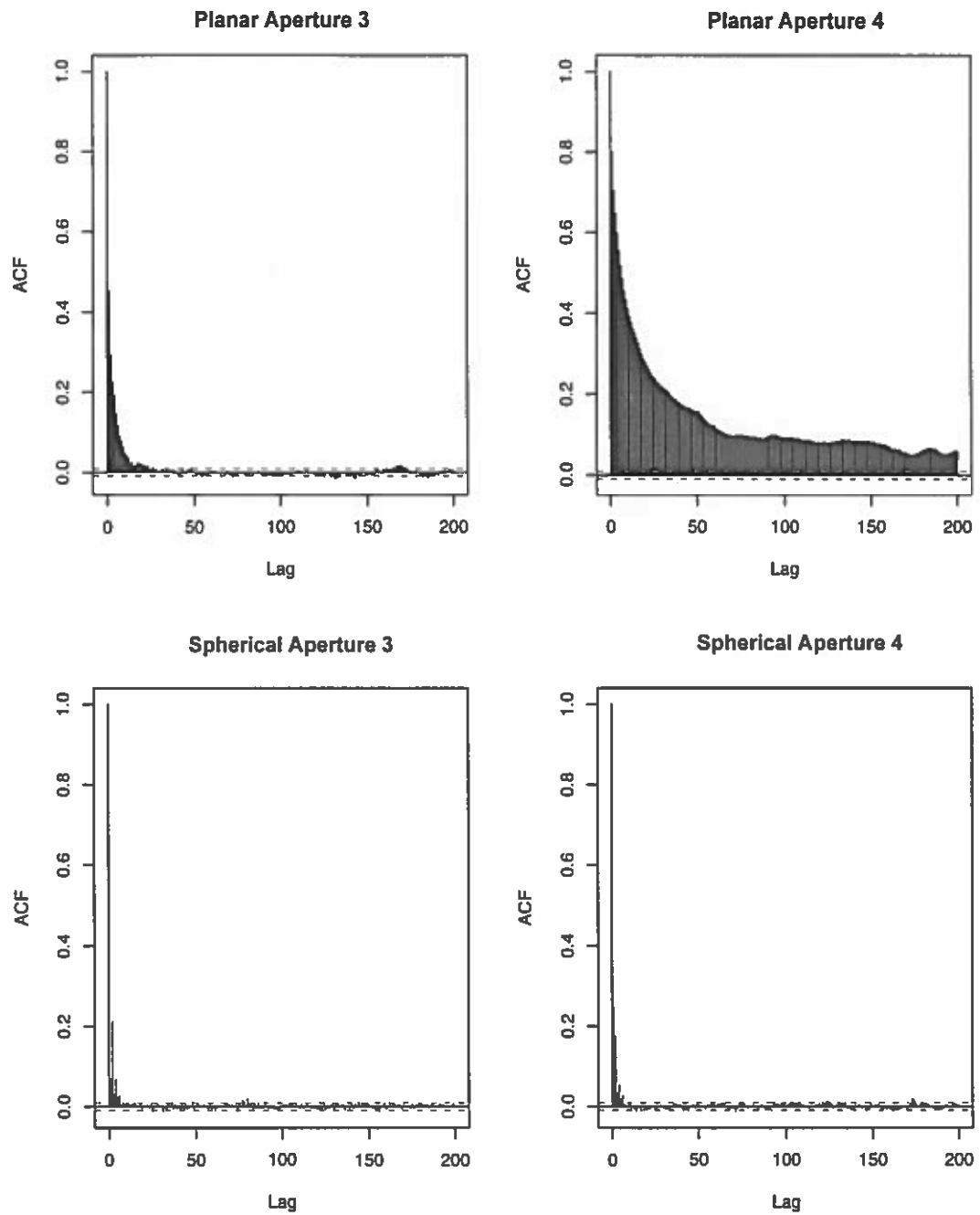


Figure 22. ACF for representative hexagon sample series. Each topology was run with four resolutions and a random 18% of cells initialized to live.

Conclusions

In this chapter we demonstrated that the nulib architecture can be used to build testbeds that can facilitate useful comparisons between DGGs alternatives.

In the current study we have implemented the first spherical multi-scale topology-independent cellular automata and conducted a preliminary set of experiments with it. These experiments appear to imply clear differences between the behavior of different spherical topologies, as well as clear differences between planar and spherical versions of the same cellular automata. In particular we note that the behavior of the planar CAs was dominated by boundary effects at the edges of the grid. But this does not occur on the closed but unbounded surface of the sphere. Instead the spherical behavior is dominated by the boundary effect of having a finite number of resolutions.

The current work is only a first step in exploring this new class of discrete simulation. In particular, a great deal of effort will be necessary to identify interesting rule sets beyond the simplistic rule set used in this study.

CHAPTER V

Location Coding on Icosahedral Aperture 3 Hexagon Discrete Global Grids

Introduction

Calculations on computer systems involving the locations of objects situated on, or referenced to, the surface of the earth require a *location coding system* — a particular computer representation of geospatial location. DGGs are a relatively new approach to the representation of location on the earth's surface. As we saw in Chapter II, DGGs have been proposed based on cells that are triangles, squares, diamonds, and hexagons. But while DGGs based on hexagons exhibit clear advantages and have become increasingly popular among many end-user communities, hexagon-based DGGs have been largely ignored by the data structures community. This is because, unlike squares, triangles, or diamonds, hexagons do not induce hierarchical data structures which are quadtrees — or even trees — and thus common tree-based algorithms cannot be directly adapted for use on hexagon DGGs.

Yet hexagon hierarchies do exhibit regularities that can be used to develop hierarchical location coding systems. Multi-resolution hexagon grids can be formed such that the center point of each resolution k cell is also a cell center point in resolution $k+1$. Such a hierarchy is known as a *central place* (Christaller, 1966) or *aligned* (Sahr et al., 2003) hierarchy. Dacey (1965) notes that aligned hexagon grids can be formed for any aperture $h^2 + hk + k^2$, where h and k are any positive integers. DGGs have been proposed using hexagon hierarchies in which increasing resolution reduces cell area by a factor of 3 ($h = 1, k = 1$) or 4 ($h = 0, k = 2$). These are known as *aperture 3* and *aperture 4* grids respec-

tively. As noted in Chapter II, these hexagon hierarchies have been tiled onto the surface of a spherical icosahedron to form DGGs. It should be noted that the sphere cannot be totally tiled using just hexagons; thus the cells of a hexagon-based icosahedral DGG always include exactly twelve pentagonal cells at each resolution; these are centered on the twelve vertices of the icosahedron. Hexagon DGGs include the Icosahedral Snyder Equal Area aperture 3 Hexagon (ISEA3H) DGG (Sahr, 2003), which has equal-area cells, and the aperture 4 hexagon DGG of Heikes and Randall (1995a, b), which has cell shapes optimized for performing global climate change simulations.

In this chapter we will first discuss the use of DGGs as data structures and the two primary approaches to developing location coding systems on multi-resolution grids. We will then propose a series of location coding systems specifically for icosahedral aperture 3 hexagon DGGs such as the ISEA3H.

Background

The cells of any DGG can form the basis of at least three types of geospatial data structure. Under the most common usage the DGG cell regions constitute the pixels of a raster system. For each resolution in a DGG, each zero-dimensional point on the earth's surface can be represented by mapping it to the DGG cell region in which it occurs. One-dimensional lines and curves can be represented as an ordered vector of cell regions intersected by the curve. A two-dimensional region can be represented using a one-dimensional representation of its boundary, or as the set of cells containing, intersecting, or contained by the region being represented.

As proposed by Dutton (1989) the center points of DGG cells can form a multi-resolution vector system. For each resolution in the DGG, each zero-dimensional point on the earth's surface is mapped to the DGG center point of the cell region in which it occurs. One-dimensional lines and curves can be represented as an ordered vector of cell point vertexes. A two-dimensional region can be represented using a one-dimensional representation of the region's boundary.

Finally, the DGGs cell regions can serve as buckets into which data objects are assigned based on their location. Depending on the application, a data object can either be assigned to the finest resolution cell which entirely contains it, or to the coarsest resolution cell which uniquely distinguishes it from all other data objects in the data set of interest. The DGGs then forms the basis for a spatial database that can be efficiently queried due to the regular geometry and hierarchical structure of the DGGs.

In order to be usable as a data structure, each cell in a DGGs must have assigned to it one or more unique location codes. On contemporary computer systems each of these location codes consists of a string of bits. A quantization operator must be defined which maps each location on the earth's surface to a single location code or set of location codes; usually the inverse mapping is also defined.

Location codes generally take one of two basic forms: pyramid addresses and path addresses. Under a pyramid address approach (Burt, 1980) each cell is assigned a unique location code within the DGG of corresponding resolution. This single-resolution location code may be multi-dimensional or linear. Given a resolution k cell with single-resolution location code k -address, we can designate the DGGs pyramid location code to be the two-tuple $(k, k$ -address). A multi-resolution DGGs address representation of a location can be constructed by taking the series of single-resolution DGG addresses ordered by increasing resolution.

Pyramid addresses are useful in applications that work with single resolution data sets. Various pyramid address location coding systems with associated quantization operators are known for grids based on triangles, squares, diamonds, and hexagons.

The resolution k quantization of a geospatial location into a DGGs cell restricts the possible resolution $k+1$ quantization cells to those whose regions overlap or are contained within the resolution k cell. We can construct a *spatial hierarchy* by designating that each resolution k cell has as children all resolution $k+1$ cells whose regions overlap or are contained within its region. A *path address* is a location code which specifies the path through a spatial hierarchy that corresponds to a multi-resolution location quantization. Path address location codes are often linear, consisting of a string of digits with each

digit in the code corresponding to a single resolution and specifying a particular child cell of the parent cell specified by the address prefix. If the numerical base of the digits is equal to the number of children at each resolution, then each possible digit value can uniquely and optimally indicate a particular child cell; in this case the path address is known as a *trie* (Fredkin, 1960).

Since pyramid addresses do not store redundant multi-resolution location information, they can be much more compact than multi-resolution pyramid addresses. And because path address location codes can be constructed such that the number of digits in the code corresponds to the maximum resolution, or precision, of the address, path addresses automatically encode their precision (Dutton, 1989), obviating the need for separate precision metadata.

Path addresses also often simplify the development of hierarchical algorithms. In particular, the coarser cell represented by the prefix of a location code can be used as a coarse filter for the proximity operations containment, equality, intersection/overlap, adjacency, and metric distance. These proximity operations are the primary forms of spatial queries used in spatial databases, and such queries are thus rendered more efficient by the use of path addresses. For example, take a bucket system where data object boundaries are assigned to the smallest containing cell. A query may ask for all data objects whose locations intersect a particular region. We can immediately discard from consideration all objects in bucket cells that do not intersect the smallest containing bucket cell of the query region (Samet, 1989).

Path addresses are usually associated with spatial hierarchies which form traditional trees, where each child has one and only one parent. The most commonly used such structure is the quadtree, consisting of a square root cell that is sub-divided into four square children, each of which is recursively sub-divided into four children, and so forth to the desired resolution. At each successive resolution one of four digits (usually 0, 1, 2, or 3) is added to the location code of the parent cell to indicate the location code of each child cell. Quadtree-like hierarchies can also be created by the four-fold recursive subdivision of triangle or diamond cells. Thus DGGSs based on the four-fold recursive

subdivision of squares, triangles, or diamonds can all be location coded as forests of quadtrees.

Figure 13 (see Chapter II) illustrates three resolutions of an aperture 3 hexagon hierarchy defined on one triangular face of an icosahedron. Note that each cell can have up to three parents, and that the spatial hierarchy induced by an aperture 3 hexagon grid is therefore not a tree. For this reason aperture 3 hexagon DGGs have previously been addressed using pyramid addressing systems. In the next section we will review one approach to pyramid addressing on aperture 3 hexagon DGGs. We will then introduce two path addressing location coding systems for these DGGs, one for use in vector data structures and one for use in raster and bucket data structures.

Pyramid Addressing on Aperture 3 Hexagon Grids: The Quadrilateral Two-Dimensional Integer System

Traditional planar cartesian coordinate systems employ two coordinate axes that are perpendicular to each other. Hexagon systems, however, naturally form three axes that are at 60° angles to each other. As illustrated in Figure 23, there are two natural orientations of these three axes relative to the traditional cartesian coordinate system, which we designate *Class I* and *Class II* as a generalization of terminology developed for triangle grid systems (Sahr et al., 2003).

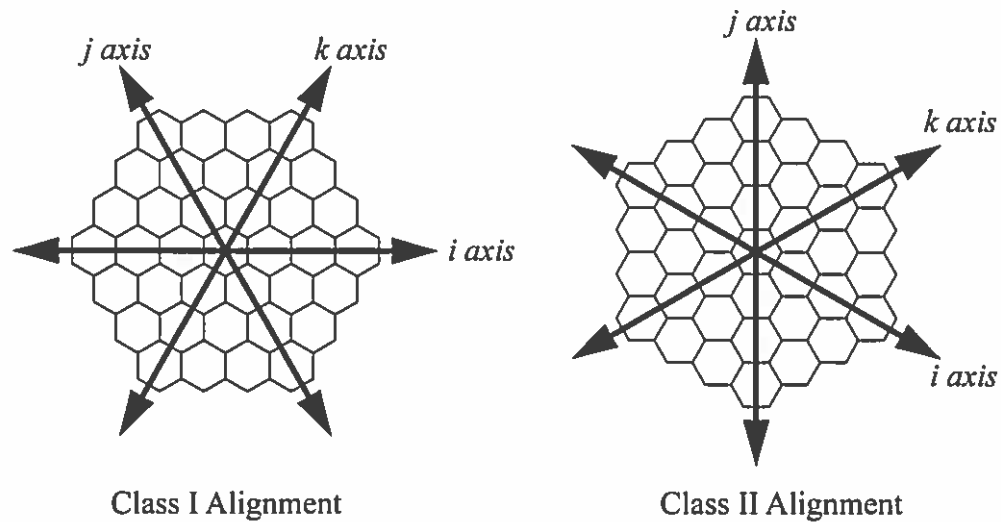


Figure 23. Three-axis hexagon coordinate systems.

Two of these axes are sufficient to uniquely identify each hexagon. Two-axis coordinate systems have been used in the development of a number of algorithms for hexagon grids, including:

1. quantization to/from cartesian coordinates (van Roessel, 1988)
2. metric distance (Luczak & Rosenfeld, 1976)
3. vector addition and subtraction (Snyder et al., 1999)
4. neighbor identification (Snyder et al., 1999)
5. adapted Bresenham's line and circle rasterization (Wuthrich & Stucki, 1991)
6. edge detection (Abu-Bakar & Green, 1996)
7. line-of-sight (Verbrugge, 1997)
8. field-of-view (Verbrugge, 1997)
9. image gradient determination (Snyder et al., 1999)
10. variable conductance diffusion (Snyder et al., 1999)

Note that, since a quantization operator is defined for two-axis coordinate sys-

tems, any alternative location coding for a hexagon grid trivially implements the other algorithms listed provided a mapping is defined between the alternative system and the two-axis coordinate system. Location codes in the alternative coding would be converted to two-axis codes, and the algorithm then performed on the two-axis codes. Any resulting location codes would be converted back to the alternative location coding.

We choose the i and j axes as a coordinate system basis because they are most useful in constructing pyramid addresses for icosahedral DGGs. We designate the resulting coordinate system a *two-dimensional integer (2di)* coordinate system. Figure 24 illustrates the assignment of coordinate addresses in a Class I 2di coordinate system.

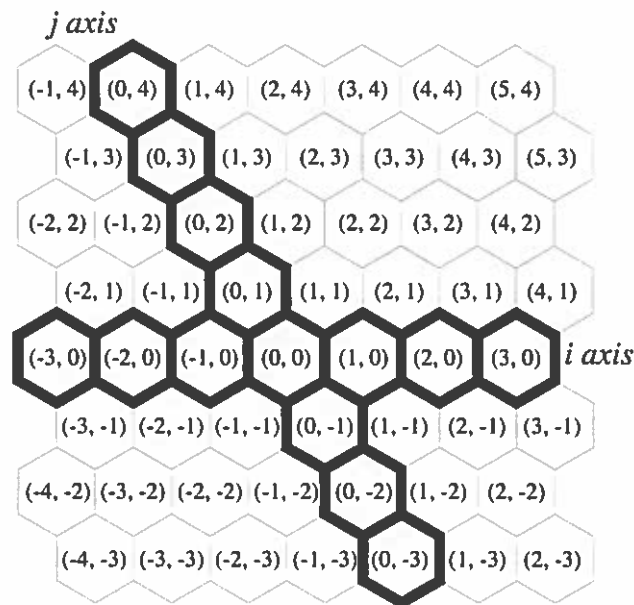


Figure 24. Class I 2di coordinate system.

A multi-resolution aperture 3 hexagon grid system may be formed as follows. Begin with a single resolution Class I hexagon grid and call it the resolution k grid. To form the next finer grid (resolution $k+1$), create a Class II hexagon grid consisting of hexes with exactly $1/3$ the area of the resolution k hexagons, with the resolution $k+1$ hexagons centered on the vertices and center points of the resolution k hexagons. Repeat this process the desired number of resolutions, alternating between Class I and Class II

grids at each successive resolution. Figure 25 illustrates four resolutions of such a grid. (Note that the series may also be started with a Class II grid, again with successive resolutions alternating class).

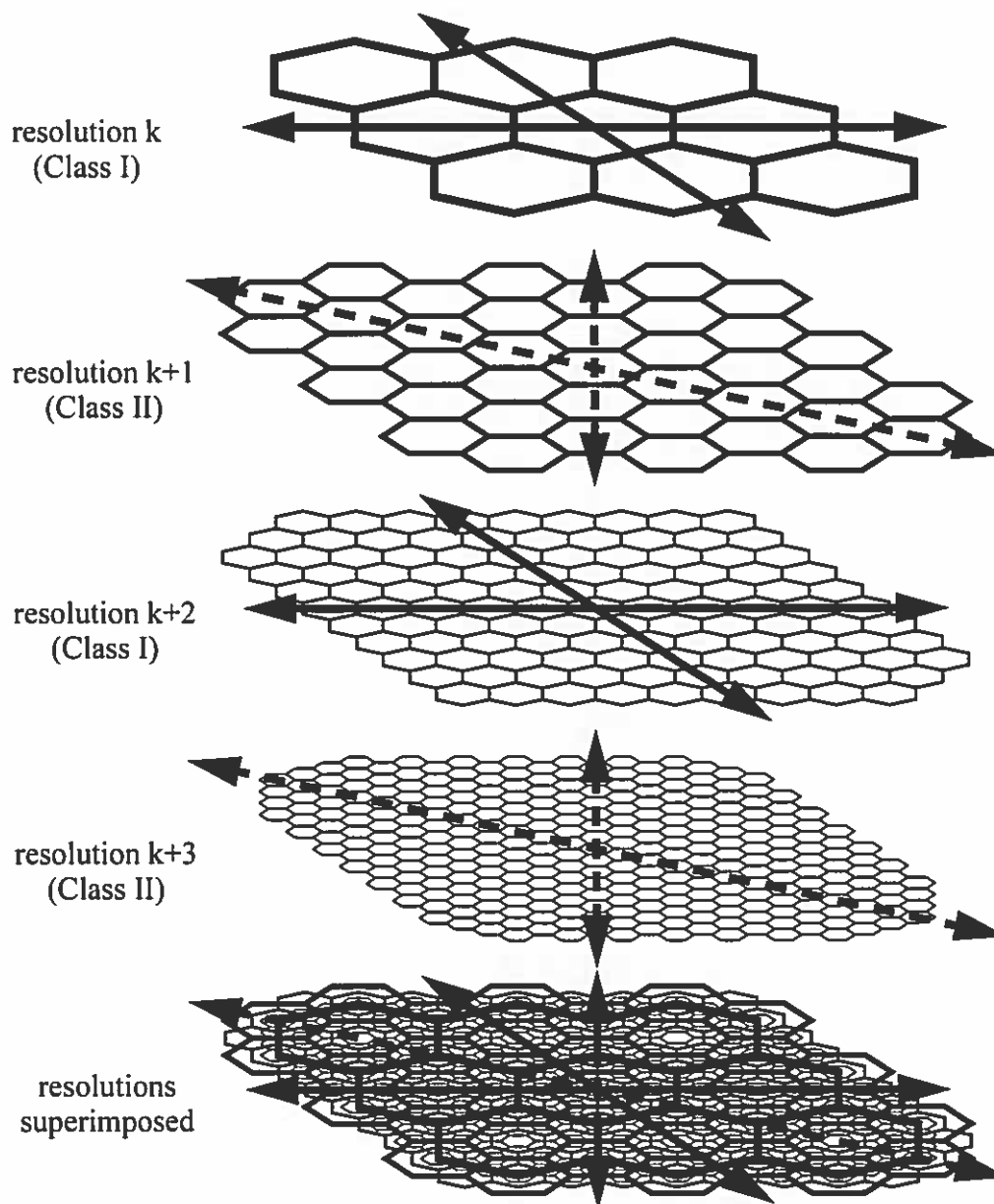


Figure 25. Four resolutions of an aperture 3 hexagon grid system.

A unique pyramid address of the form $[r, (i, j)]$ may be assigned to each hexagon in the multi-resolution grid, where r is the resolution of the hexagon and (i, j) is the 2di address of the hexagon on the resolution r grid.

The twenty triangular faces of the icosahedron can be paired into ten quadrilaterals as illustrated in Figure 26. Each of these quadrilaterals can be addressed using a 2di coordinate system. Each 2di coordinate system origin corresponds to one of the pentagonal vertex cells. Two pentagonal vertex cells are left-over and can be treated as single-cell 2di coordinate systems at every resolution. Figure 27 shows 2di coordinate systems on an icosahedron unfolded onto the plane and with the quadrilaterals given one possible numbering. The pentagonal vertex cells have been drawn as hexagons.

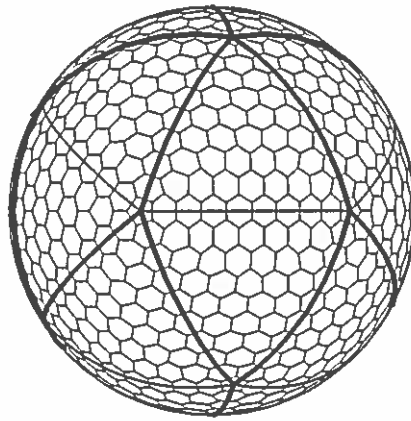


Figure 26. Icosahedral faces paired to form ten quadrilaterals.

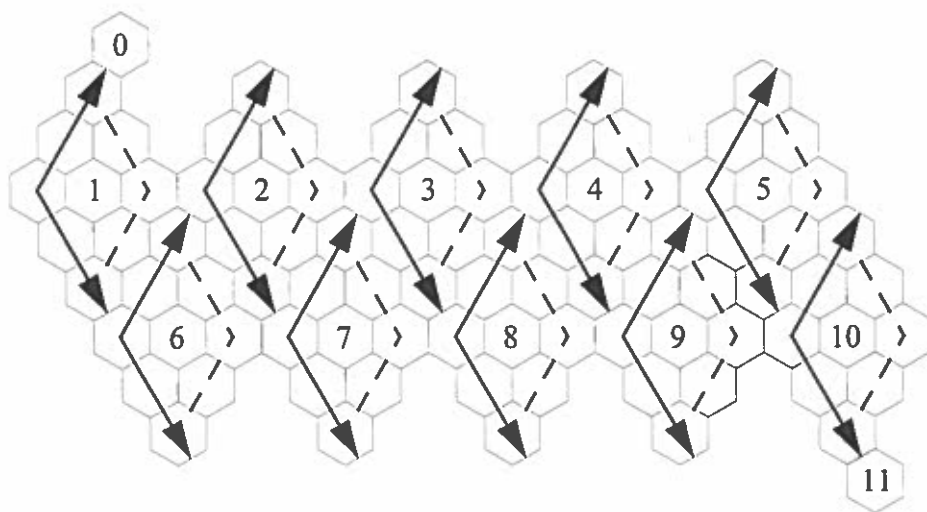


Figure 27. Unfolded icosahedron with 2di coordinate systems.

Class II grids do not conveniently align with the Class I coordinate axes naturally defined by the quadrilateral edges. We note that for every cell of a Class II resolution k grid there is a single Class I resolution $k+1$ cell centered on that resolution k cell. Thus without ambiguity we can assign to each resolution k cell the coordinates of the class I resolution $k+1$ cell centered upon it. Figure 28 illustrates one Class II resolution quadrilateral addressed using this method.

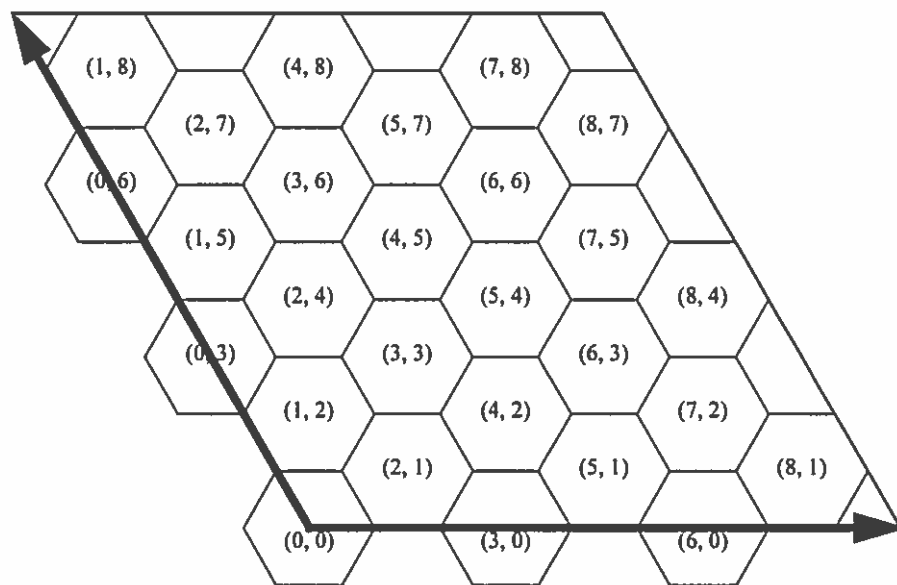


Figure 28. Class II grid on a single quadrilateral addressed using underlying Class I coordinates.

A unique pyramid address of the form $\{r, [q, (i,j)]\}$ may be assigned to each cell in the multi-resolution grid, where r is the resolution of the hexagon, q is the quadrilateral on which the cell occurs, and (i, j) is the 2di address of the hexagon on the quadrilateral q resolution r grid. This icosahedral coordinate system is designated the *quadrilateral 2di (q2di) system* (Sahr, 2002).

Path Addressing on Aperture 3 Hexagon Grids

Vector Location Coding: Icosahedral Modified Generalized Balanced Ternary

Let (x, y) be a zero-dimensional point location on the plane. Then a resolution k aperture 3 hexagon grid quantization of this point restricts the possible resolution $k+1$ quantization of the point to the seven resolution $k+1$ hexagons that overlap the resolution k hexagon. This is illustrated in Figure 29.

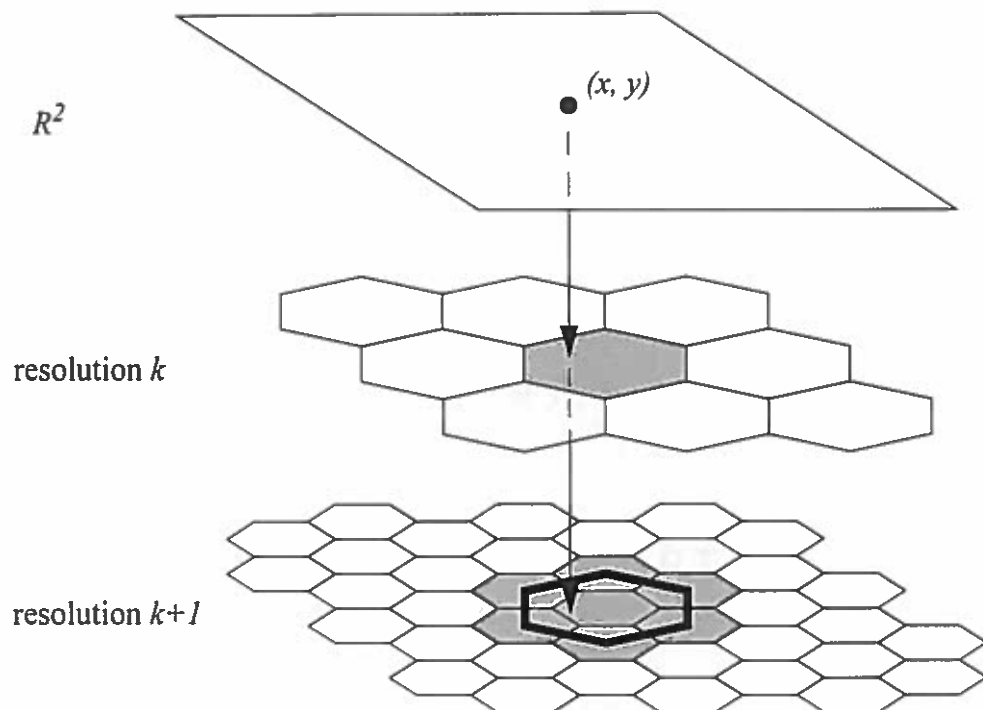


Figure 29. Quantization at one resolution restricts possible quantization candidates at higher resolutions.

Given a linear code for a resolution k cell in a planar aperture 3 hexagon grid, a multi-scale hierarchical coding scheme can be specified by assigning specific digits to each of the seven possible resolution $k+1$ cells, and then applying this scheme iteratively until the desired maximum resolution is achieved.

An address can be assigned in a fashion inspired by the Generalized Balanced Ternary (GBT) system (Gibson & Lucas, 1982) for hexagon aggregation and thus we call

this system Modified GBT (MGBT). Figures 30 and 31 show the assignment of resolution $k+1$ address digits based on class I/class II resolution k addresses respectively.

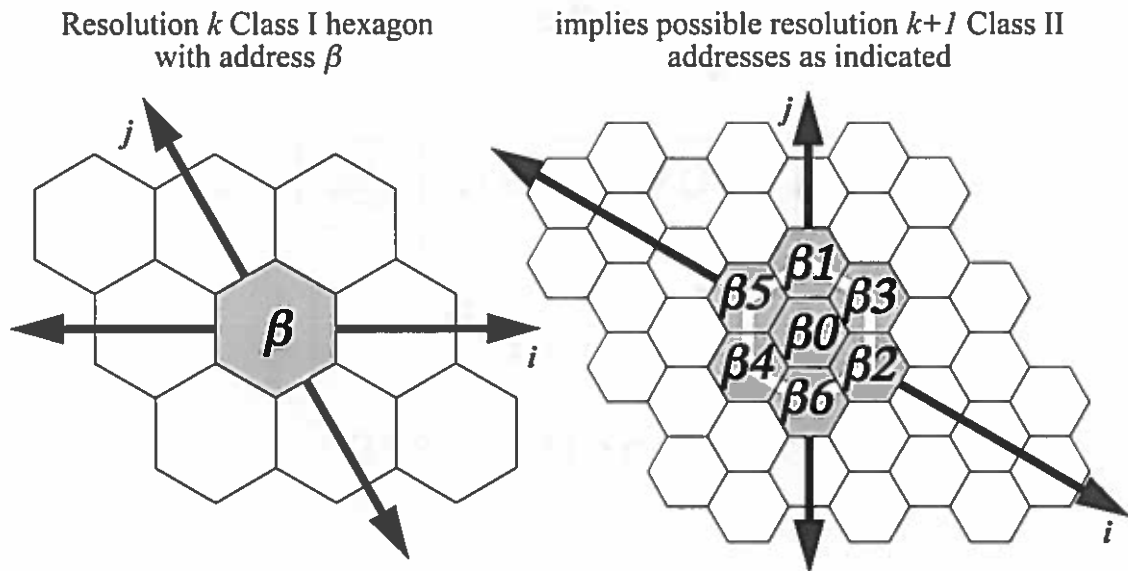


Figure 30. MGBT addressing of Class II children.

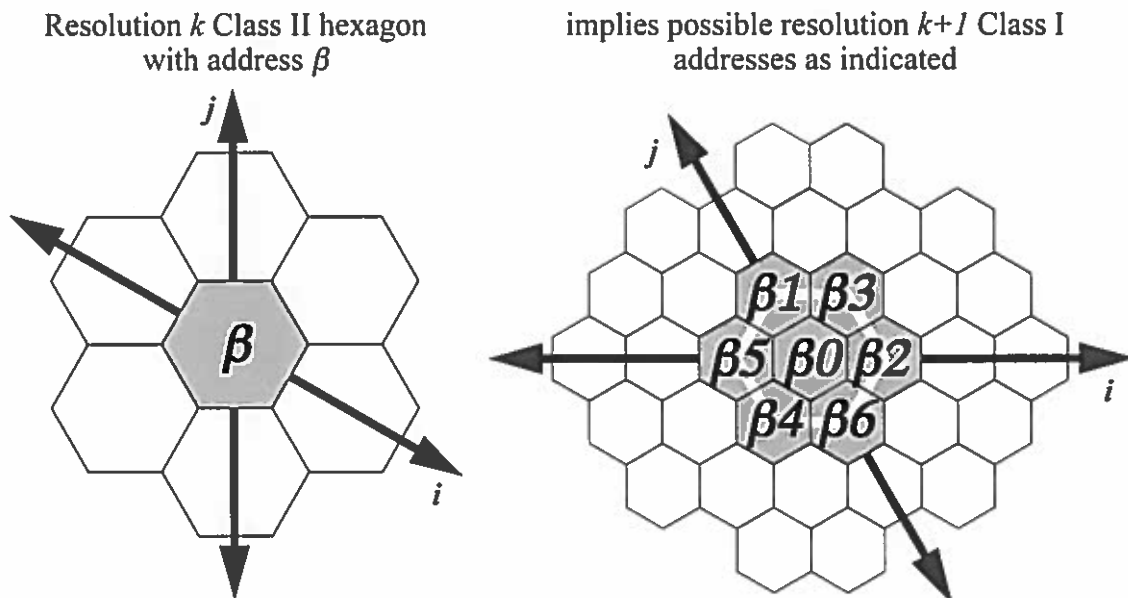


Figure 31. MGBT addressing of Class I children.

Each resolution $k+1$ hexagon may be overlapped by up to three different resolution k parents to which different regions of it belong. This means that MGBT effectively addresses sub-regions of hexagons, such that each entire hexagon is not assigned a unique

location code. Figure 32 illustrates the sub-regions that are addressed by three Class I hexagons.

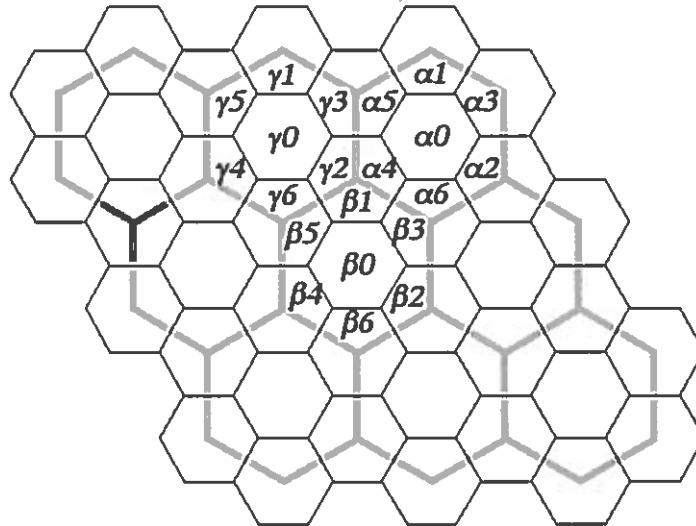


Figure 32. Sub-regions addressed by MGBT.

Since at each resolution one of seven digits is added to the location code, it is possible for each digit to be represented using three bits. Note that three bits can represent eight distinct digits. The decimal digits 0-6 have been assigned as illustrated in Figures 23 and 24 above. The remaining eighth digit, decimal 7 or binary 111, can serve a number of useful functions. If the addresses are variable length, then a 7-digit can be concatenated to an address to indicate address termination. If the addresses are fixed length, a 7-digit can be used to indicate that the remaining higher resolution digits are all center digits (i.e., zero), and therefore that additional resolution will not add information to the location.

We can extend planar MGBT to a DGGs location coding by tiling an icosahedral aperture 3 hexagon DGGs with MGBT tiles. Tiles centered on the twelve icosahedral vertices form pentagons and thus require special tiling units. These can be constructed by deleting one-sixth of the sub-hierarchy generated in the hexagon case. These are coded using the procedure outlined for hexagonal MGBT tiles except that pentagonal tiles have a single sub-digit sequence deleted. That is, for pentagonal tiles with address base address

A , all sub-tiles are indexed as per the corresponding MGBT indexing except that sub-tiles with sub-indexes of the form AZd are not generated, where Z is a string of 0 or more zeroes and d is the sub-digit sequence (1, 2, 3, 4, 5, or 6) chosen for deletion. All hierarchical descendants of such tiles are likewise not indexed. We use $MGBT-d$ to indicate an MGBT tile with sub-digit sequence d deleted (e.g., MGBT-2 would indicate an MGBT tile with sub-digit sequence 2 chosen for deletion).

Given hexagonal and pentagonal MGBT tiles, we may tile the icosahedron to create the icosahedral MGBT (iMGBT) location coding system. In order to fully specify a geospatial coding system based on the iMGBT, a fixed orientation must be specified for each tile. One approach to achieving this is to unfold the icosahedron onto the plane and then specify that each tile be oriented consistently with this planer tiling. Figure 33 shows one such orientation. Each base tile is labeled with a base tile number. An iMGBT location code can have one of the following four easily inter-convertible forms:

1. *Character string code form.* The location code consists of a string of digits beginning with the two-digit base tile code (00, 01, 02, ... 31) followed by the digit string corresponding to the appropriate address of each finer resolution within the tile.
2. *Integer code form.* The character string code form may be interpreted and stored as a single integer value.
3. *Modified integer code form.* When displaying integer values leading zeroes are usually removed. This will result in differences in the number of digits between codes on base tiles 00-09 and those of the same resolution but on other base tiles. This can be remedied by adding the value of 40 (four being the lowest unused value for the 10's digit) to each of the base tile values so that they are numbered 40, 41, ... 71. Note that since digits 4-7 are not used as leading digits in the integer code form, modified integer codes can be unambiguously distinguished from the standard integer codes.
4. *Packed code form.* Under this form, the base tile codes are stored as five-digit binary numbers. Sub-codes within each tile are stored as a packed series of three-bit binary digits (as described above) and appended to the base tile number to fully specify a geospatial code. Under this scheme, codes up to resolution 10 can be stored in 32 bits of

contiguous storage, and codes up to resolution 20 can be stored in 64 bits of contiguous storage.

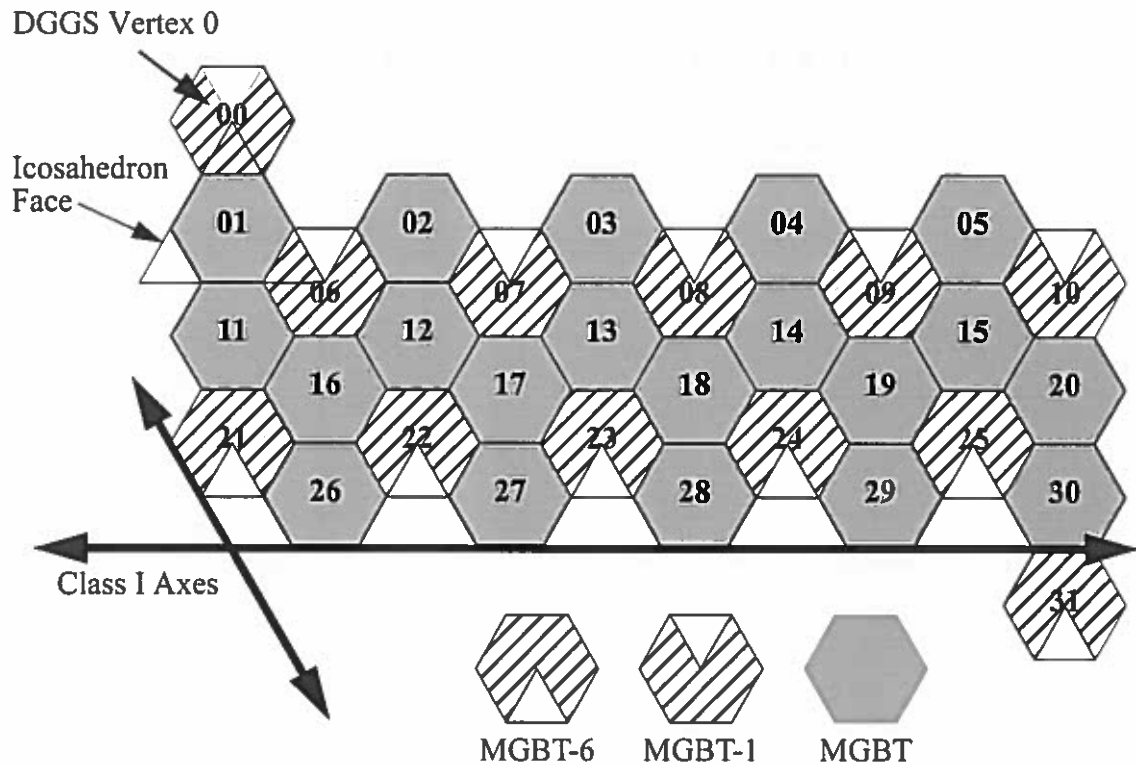


Figure 33. iMGBT base tiling on an unfolded icosahedron.

An iMGBT code for a point location can be truncated such that any prefix of the code yields a valid quantification of the point location at a coarser grid resolution. This allows prefixes of the code to be used as a coarse filter for the proximity operations equality, adjacency, and metric distance.

Raster Location Coding: The Icosahedral Aperture 3 Hexagon Tree

As mentioned above, the iMGBT effectively addresses sub-regions of cells and thus assigns multiple codes to many of the cells in an aperture 3 hexagon DGGs. This is reasonable if the DGGs is to be used as a vector system, since the primary data objects are zero-dimensional points that do indeed map to hexagon sub-regions. But in the case

of raster or bucket systems data objects are mapped to entire cells and it is therefore useful to have a specific unique location code for each cell. If we wish to map the cells to physical resources — to perform parallel processing, for instance — the assignment of a unique code becomes a necessity.

For applications where unique cell location codes are required we must decompose the iMGBT into unique sub-trees while, hopefully, retaining the regular hexagonal topology of the underlying DGGS. No satisfactory way of achieving this is known using a consistent sub-tree topology. However, if we allow the decomposition to consist of two sub-tree topologies a solution is given by employing the planar *Aperture 3 Hexagon Tree* (A3HT) (Sahr, Peterson, and Lutterodt, 2003).

Each hexagon in an A3HT is assigned one of two generator hexagon types: *open* (type *A*), or *closed* (type *B*). As illustrated in Figure 34, an open generator at resolution k generates a single resolution $k+1$ hexagon that is a closed generator centered on itself. A closed generator hexagon at resolution k also generates a single resolution $k+1$ closed generator hexagon at its center. The closed generator in addition generates six resolution $k+1$ open generator hexagons, one centered at each of its six vertices. Figure 35 illustrates a closed generator.

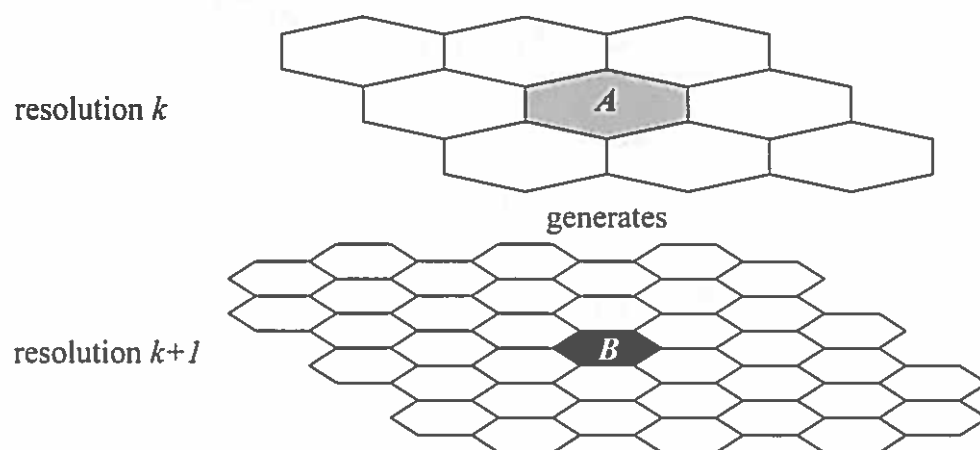


Figure 34. A3HT open (type A) generator hexagon.

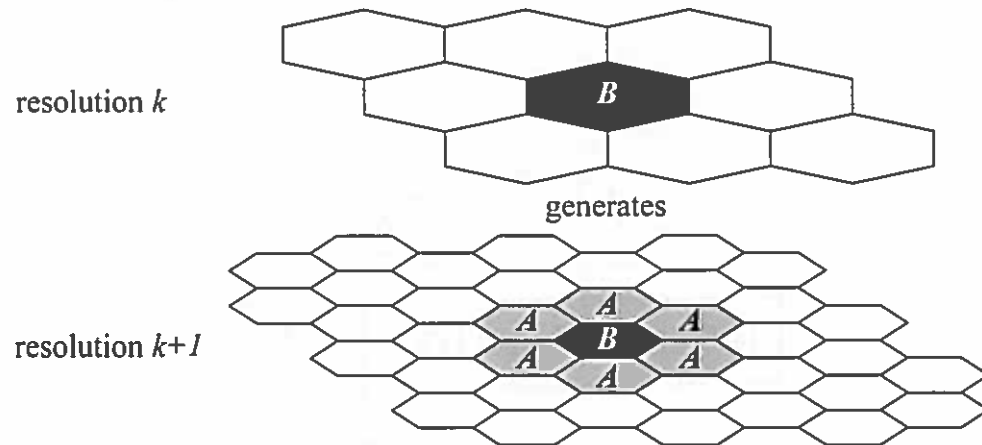


Figure 35. A3HT closed (type B) generator hexagon.

An A3HT of arbitrary resolution can be created by beginning with a single open or closed hexagon and then recursively applying the above generator rules until the desired resolution is reached. Figure 36 shows the first four resolutions of an A3HT generated by a resolution k closed generator hexagon.

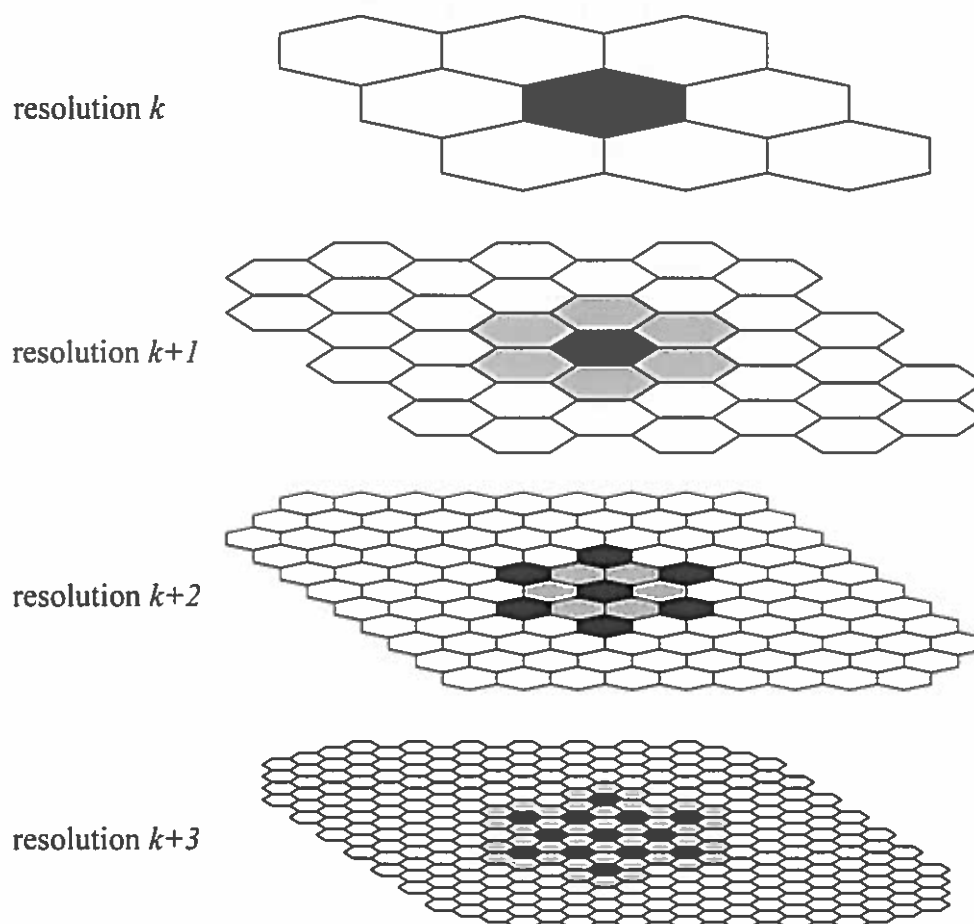


Figure 36. Generation of four resolutions of an A3HT grid from a closed (type B) initial generator.

Hexagons in an A3HT may be addressed using the corresponding MGBT codes. In all cases (Class I or Class II, open or closed generator), the address of centroid children are formed by concatenating a zero digit with the parent hexagon address. The addresses of vertex children of closed generators are formed by concatenating one of the digits 1-6 with the parent hexagon address, using the MGBT addressing given for Class I and Class II parents in Figures 30 and 31 respectively.

We can use the A3HT to address an icosahedral aperture 3 DGGs by tiling the DGGs with A3HT tiles. As in the case of the iMGBT, tiles centered on the twelve icosahedral vertices form pentagons and thus require special tiling units.

Aperture 3 Pentagon Tree (A3PT) tiling units are generated similarly to A3HT tiles except that one-sixth of the sub-hierarchy is deleted. Each pentagon in an A3PT is assigned one of two generator pentagon types: open (type A) or closed (type B). As illustrated in Figure 37, an open A3PT generator at resolution k generates a single resolution $k+1$ pentagon that is a closed generator centered on itself. A closed A3PT generator at resolution k also generates a single resolution $k+1$ closed A3PT generator pentagon at its center, but in addition it generates five resolution $k+1$ open A3HT generator hexagons, one centered at each of its five vertices. A closed A3PT generator is illustrated in Figure 38. An A3PT of arbitrary resolution can be created by beginning with a single open or closed A3PT and then recursively applying the above generator rules until the desired resolution is reached.

Like the MGBT- d tiles described in the previous section, A3PT tiles are addressed with a single sub-digit sequence deleted. $A3PT-d$ is used to indicate an A3PT tile with the sub-digit sequence d deleted. This deletion occurs exactly as per the corresponding MGBT- d tile.

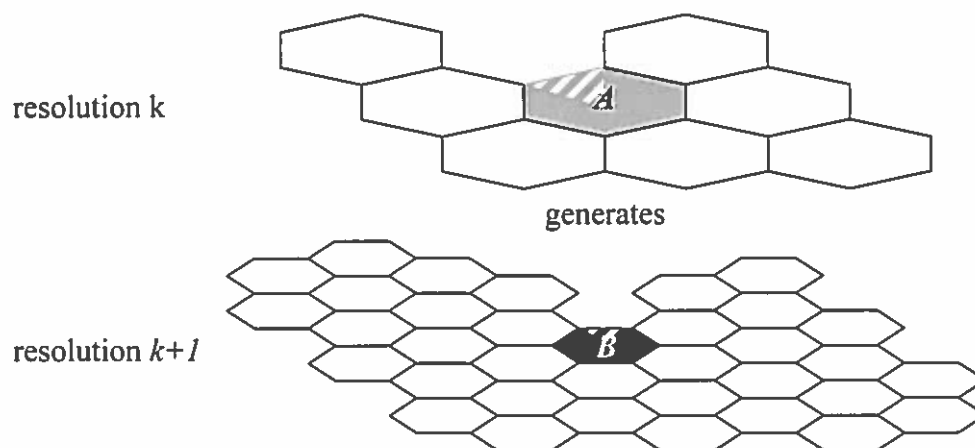


Figure 37. A3PT open (type A) generator unfolded on the plane.

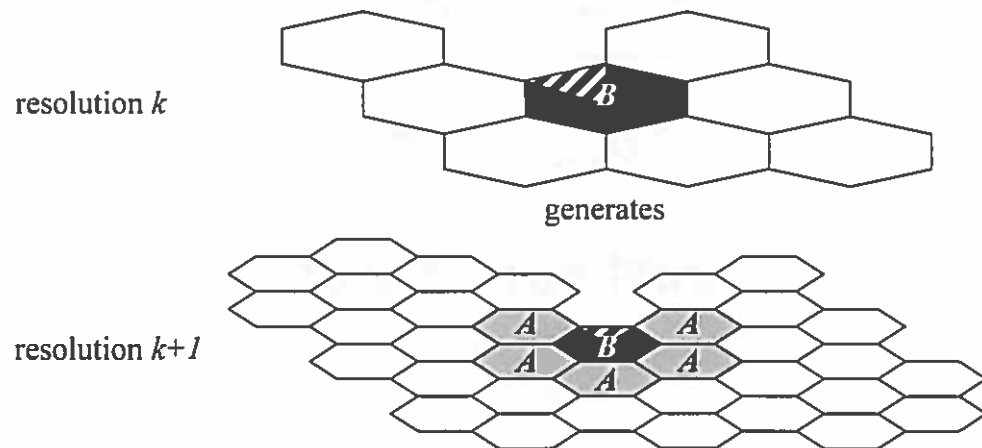


Figure 38. A3PT closed (type B) generator unfolded on the plane.

Given A3HT and A3PT tiles, we may tile the icosahedron to create the *icosahedral A3HT* (iA3HT). As with the iMGBT we can specify the orientation of base tiles by orienting them consistently with a planar unfolding of the icosahedron. In this case we must also indicate which base tiles are closed and which are open. Figure 39 illustrates a base tiling for the iA3HT with tiles labeled with base location codes. Figure 40 illustrates the first six resolutions of an A3HT generation pattern on the ISEA3H DGGS.

The iA3HT can be indexed using any of the four location code forms given for the iMGBT in the last section. We note that since each open A3HT or A3PT generator generates only a single center hexagon, every non-zero digit in an iA3HT location code must be followed by a 0 digit. A first order compression of the codes can be achieved by eliminating these redundant 0 digits.

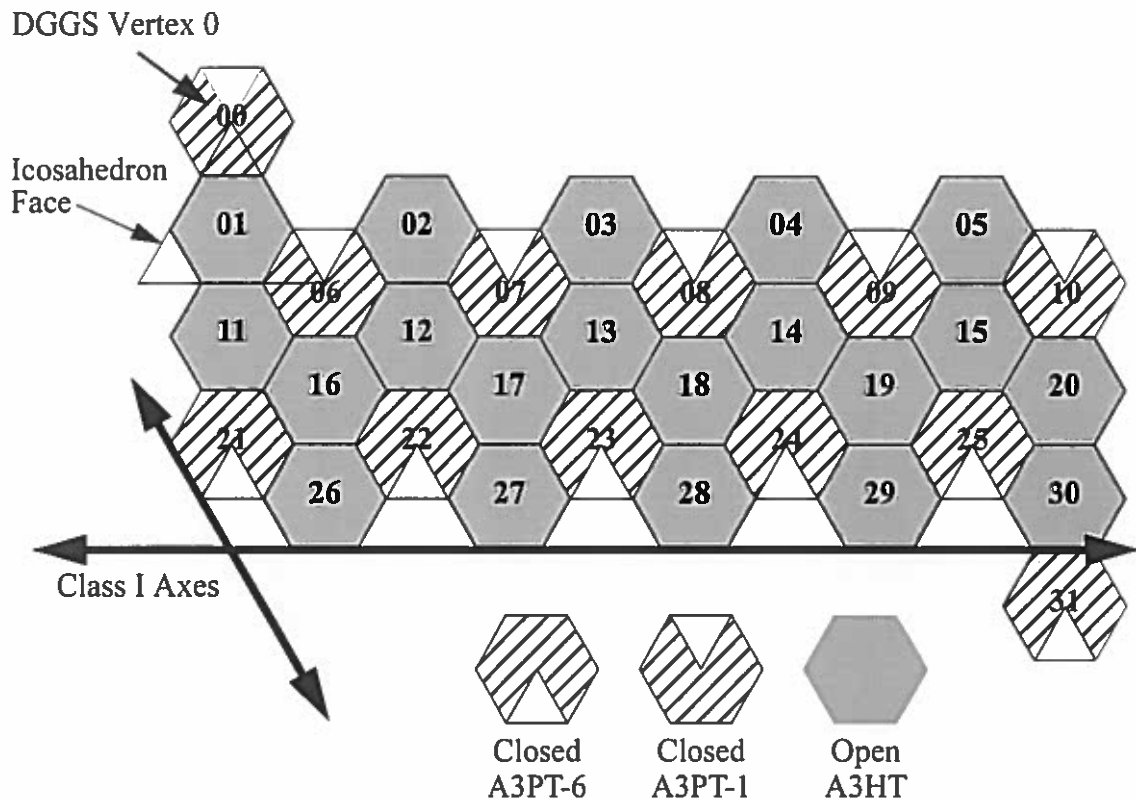


Figure 39. iA3HT base tiling on an unfolded icosahedron.

The iA3HT system provides a unique location code for each cell in an icosahedral aperture 3 hexagonal DGGs with many of the desirable characteristics of traditional tree-based path addresses. Since each resolution (beyond the base tiling) is encoded using a single digit, the codes implicitly encode precision metadata. Further, any prefix of an iA3HT location code is guaranteed to be a valid cell at a correspondingly coarser resolution. But because closed iA3HT generators are prefixes for child cells which they do not entirely cover, they cannot be used as a coarse filter for all of their child cells for the proximity operations equality, adjacency, and metric distance. However, we note that all children of a closed iA3HT generator are fully contained in the parent of that generator. Thus in all cases prefixes two resolutions coarser than the target resolution can be used as coarse filters for proximity operations.

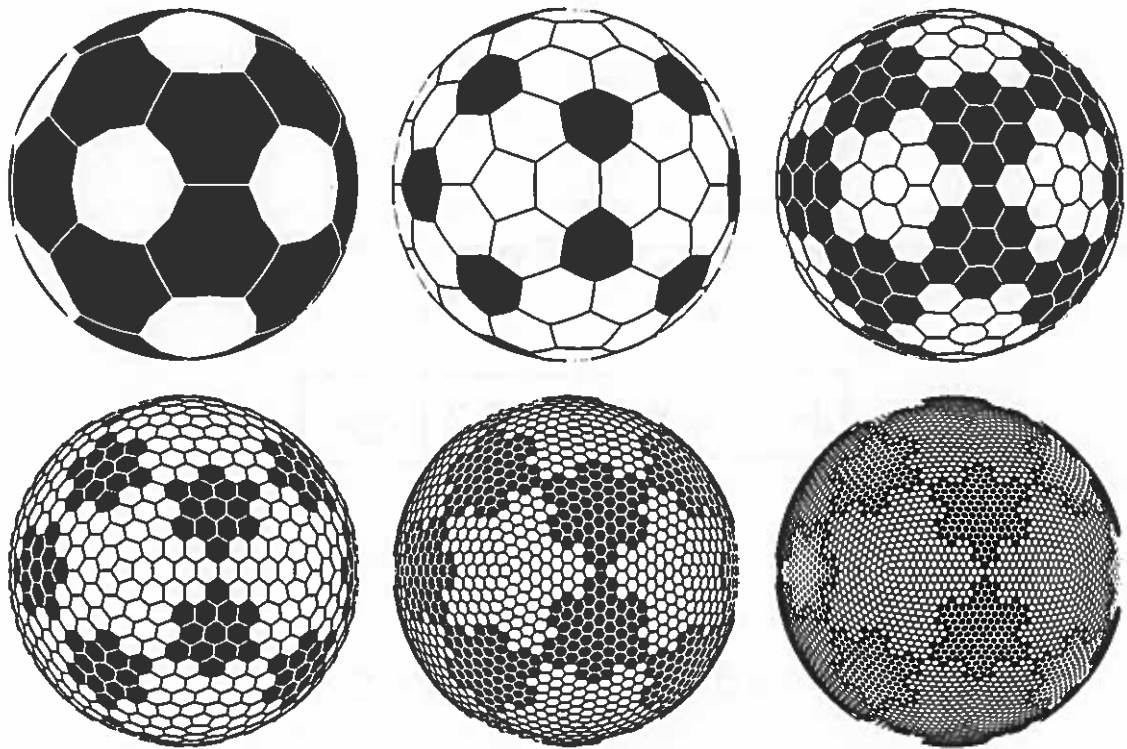


Figure 40. The first six resolutions of an iA3HT generation pattern on the ISEA3H DGGs. The shading indicates the sub-trees generated by the base tiles.

Quantization Algorithm for the iA3HT

In order to use the iA3HT as a location coding system we must define a quantization operator that maps locations on the earth's surface to iA3HT location codes. As previously discussed such an algorithm has been developed for the q2di coordinate system; thus we need only develop an algorithm that maps q2di coordinates to iA3HT location codes. In the next section we give algorithms for conversions between planar A3HT and 2di pyramid systems. In the following section we then extend these algorithms to icosahedral DGGs by defining the conversion between q2di coordinates and iA3HT codes.

A3HT Quantization Algorithms

The definitions that follow assume that the root (resolution 0 hexagon) of all A3HT indices are Class I type A generators. Algorithm definitions for initial generators that are Class II and/or type B follow trivially from the algorithms given below. The existence of the following algorithms are assumed; all are trivially definable based on the A3HT definition given above except for *distance*, which is given in Luczak & Rosenfeld (1976):

digit:A3HT x integer → *integer*. Returns a specific resolution digit from an A3HT index.

distance:2di x 2di → *integer*. Returns the metric distance between two 2di addresses.

generateNextLevel:A3HT → {*A3HT*}. Returns the set of child location codes generated by an A3HT location code.

isClassI:A3HT → *boolean*. Returns whether or not an A3HT index is Class I.

isTypeA:A3HT → *boolean*. Returns whether or not an A3HT index is type A.

resolution:A3HT → *integer*. Returns the resolution of an A3HT index.

resolve:A3HT x integer → *A3HT*. Extend an A3HT index to a specified resolution by padding it with trailing 0's.

Conversion from A3HT to 2di pyramid coordinates

The algorithm *a3htToCoord2di* gives a 2di address corresponding to a specified A3HT location code. Algorithms used within this definition are given following the main algorithm definition.

ALGORITHM *a3htToCoord2di:A3HT* → *2di*

Convert A3HT index *ndx* to a 2di address *coord* of the same resolution as *ndx*.

BEGIN *a3ht2Coord2di*

coord ← (0, 0)
sameClass ← true

```

resOffset ← 0
for i = resolution(ndx) to 0 step by -1

  if sameClass
    coord ← coord + sameClassDownN(digit(ndx, i), resOffset)
  else
    if (isClassI(ndx))
      coord ← coord + classIIdownOne(digit(ndx, i)) *
        3.0(resOffset-1)/2
    else
      coord ← coord + classIdownOne(digit(ndx, i)) *
        3.0(resOffset-1)/2
    end if-else
  end if-else

  sameClass ← not sameClass
  resOffset ← resOffset + 1

end for

return coord

```

END a3htToCoord2di

ALGORITHM sameClassDownN:integer x integer → 2di

Given an A3HT digit **digit** calculate the corresponding 2di **coord** on the grid **n** resolutions finer than the grid on which the A3HT digit is defined. Assumes both grids are of the same class.

BEGIN sameClassDownN

```

if digit = 0
  coord ← (0, 0)
else if digit = 1
  coord ← (0, 1)
else if digit = 2
  coord ← (1, 0)
else if digit = 3
  coord ← (1, 1)
else if digit = 4
  coord ← (-1, -1)
else if digit = 5
  coord ← (-1, 0)
else if digit = 6
  coord ← (0, -1)
end if-else

```

```

coord ← coord * 3.0n/2
return coord

```

END sameClassDownN

ALGORITHM classIIdownOne:integer → 2di

Given an A3HT digit **digit** defined on a Class II grid calculate the corresponding 2di **coord** on the Class I grid one resolution finer.

BEGIN classIIdownOne

```

if digit = 0
  coord ← (0, 0)
else if digit = 1
  coord ← (1, 2)
else if digit = 2
  coord ← (1, -1)
else if digit = 3
  coord ← (2, 1)
else if digit = 4
  coord ← (-2, -1)
else if digit = 5
  coord ← (-1, 1)
else if digit = 6
  coord ← (-1, -2)
end if-else

```

```

return coord

```

END classIIdownOne

ALGORITHM classIdownOne:integer → 2di

Given an A3HT digit **digit** defined on a Class I grid calculate the corresponding 2di **coord** on the Class II grid one resolution finer.

BEGIN classIdownOne

```

if digit = 0
  coord ← (0, 0)
else if digit = 1
  coord ← (-1, 1)
else if digit = 2
  coord ← (2, 1)
else if digit = 3
  coord ← (1, 2)

```



```

else if digit = 4
    coord ← (-1, -2)
else if digit = 5
    coord ← (-2, -1)
else if digit = 6
    coord ← (1, -1)
end if-else

return coord

```

END classIdownOne

Conversion from 2di pyramid coordinates to A3HT

The algorithm *coord2diToA3HT* gives an A3HT index corresponding to a 2di pyramid address. Algorithms used within this definition are given following the definition.

ALGORITHM *coord2diToA3HT*: 2di x integer → A3HT

Convert 2di address **coord** of resolution **finalRes** to an A3HT index **ndx**.

BEGIN *coord2diToA3HT*

```

if finalRes = 0
    return 0
end if

if finalRes = 1
    return 00
end if

oldSet ← null set
newSet ← generateNextLevel(00)
for res = 2 to finalRes - 1 step by 1

    oldSet ← newSet
    newSet ← null set

    for each ndx in oldSet
        if confirmedDescendent(ndx, coord, res)
            newSet ← generateNextLevel(ndx)
            break for
        else if possibleDescendent(ndx, coord, res)
            newSet ← newSet + generateNextLevel(ndx)
        end else-if
    end for
end for

```

```

end for

for each ndx in newSet
  if coord = a3ht2Coord2di(ndx)
    return ndx
  end if
end for

```

END coord2diToA3HT

ALGORITHM confirmedDescendent:A3HT x 2di x integer → boolean

Determine whether or not the 2di address **coord** of resolution **res** is definitely a descendent of the A3HT index **ndx**. Note that the vector **radius** used in this algorithm is empirically derived.

BEGIN confirmedDescendent

```

radius ← { 0, 0, 1, 1, 3, 4, 9, 13, 27, 40, 81, 121, 243, 364,
           729, 1093, 2187, 3280, 6561, 9841, 19683, 29524, 59049, 88573,
           177147, 265720, 531441, 797161, 1594323, 2391484 }

if isTypeA(ndx)
  resDiff ← res - resolution(ndx)
else
  resDiff ← res - resolution(ndx) + 1
end if-else

if distance(coord, a3ht2Coord2di(resolve(ndx, res))) >
  radius(resDiff)
  return false
else
  return true
end if-else

```

END confirmedDescendent

ALGORITHM possibleDescendent:A3HT x 2di x integer → boolean

Determine whether or not the 2di address **coord** of resolution **res** is possibly a descendent of the A3HT index **ndx**. Note that the vector **radius** used in this algorithm is empirically derived.

BEGIN possibleDescendent

```

radius ← { 0, 0, 1, 2, 4, 8, 13, 26, 40, 80, 121, 242, 364, 728,

```

```

1093, 2186, 3280, 6560, 9841, 19682, 29524, 59048, 88573,
177146, 265720, 531440, 797161, 1594322, 2391484, 4782968 }

if isTypeA(ndx)
  resDiff ← res - resolution(ndx)
else
  resDiff ← res - resolution(ndx) + 1
end if-else

if distance(coord, a3ht2Coord2di(resolve(ndx, res))) >
  radius(resDiff)
  return false
else
  return true
end if-else

```

END possibleDescendent

A3HT 60° rotation algorithm

In addition to the algorithms for conversion from 2di coordinates to/from A3HT codes, our iA3HT conversion algorithm will require an algorithm for rotating A3HT codes in 60° increments. Rotation of an A3HT cell 60° counter-clockwise about any prefix cell is performed by making the substitutions given in Table 7 for all digits in the cell code following the prefix code.

For example, the cell A0506 (where A is the base cell code) rotated 60° counter-clockwise about the prefix A would have a new code of A0402.

This algorithm can be efficiently implemented by creating separate tables for rotations of 60°, 120°, 180°, 240°, and 300°, clockwise and counter-clockwise, through multiple application of the table above. Further efficiency can be gained by generating tables for multiple digit sub-sequences, rather than for single digits alone.

Table 7. Substitutions for rotation of an A3HT cell 60° counter-clockwise about any prefix cell.

Original Digit	Rotated Digit
0	0
1	5
2	3
3	1
4	6
5	4
6	2

iA3HT Quantization Algorithms

We can now define an algorithm for conversion from q2di coordinates to iA3HT location codes.

First note that the iA3HT base cells pictured in Figure 32 correspond to resolution 1 of the q2di system. Thus q2di resolution 0 (which consists of just the 12 pentagonal vertices) has no corresponding representation under iA3HT. Also note that the Class I axes of the q2di system illustrated in Figure 27 are rotated 60° clockwise relative to the iA3HT Class I axes shown in Figure 32.

Let A^n be the resolution n iA3HT code consisting of the prefix code A followed by $n - 1$ zero's. For example, the iA3HT code 0600000 could be written 06^6 . Note that all q2di coordinates of the form $\{r, [q, (0, 0)]\}$ correspond to vertices of the icosahedron and map directly to iA3HT coordinates A' , where A is the iA3HT base code corresponding to the origin of the quadrilateral q . Table 8 lists the corresponding iA3HT base code

for each quadrilateral. Note that since quadrilaterals 0 and 11 consist of only a single origin cell at all resolutions, this table fully defines the transformation from these q2di quadrilaterals to iA3HT.

Table 8. Quadrilateral q vs. iA3HT base index A .

q	0	1	2	3	4	5	6	7	8	9	10	11
A	00	10	06	07	08	09	21	22	23	24	25	31

Temporarily ignoring the complications introduced by the pentagon cells, it is useful to note that we can view each quadrilateral as lying on a closed A3HT generator two resolutions coarser than the base iA3HT tiles. Figure 41 illustrates this relationship. Assuming the base closed A3HT generator has address X , Figure 41 gives the corresponding codes for all A3HT cells coincident with the quadrilateral.

The iA3HT resolution 0 image in Figure 41 provides the key to transforming from q2di to iA3HT codes. The basic steps for transforming a non-origin Class I q2di coordinate $\{r, [q, (i, j)]\}$ to iA3HT are as follows:

1. Perform a transformation (using *coord2diToA3HT* as defined in the previous section) from 2di pyramid coordinate $[r+1, (i, j)]$ to the corresponding A3HT code in the system defined by the Class II closed generator X (as shown in Figure 41). Assume that the 2di axes are aligned with the A3HT Class I axes.
2. Rotate the A3HT address 60° clock-wise to compensate for the q2di axes offset.
3. Replace the first 3 digits of the address with the corresponding iA3HT base cell code found in Table 9.

For most cells the resulting address will be the correct iA3HT address. In two cases additional adjustments are necessary. These adjustments require that the cell in question be rotated in 60° increments about its center-point. This transformation can be accomplished by translating the cell center to the origin, performing the rotation, and then translating the cell center back to its original position. In the case of A3HT cells the same effect can be achieved by applying the rotation algorithm to all digits except the base cell

code prefix. We call this a sub-rotation.

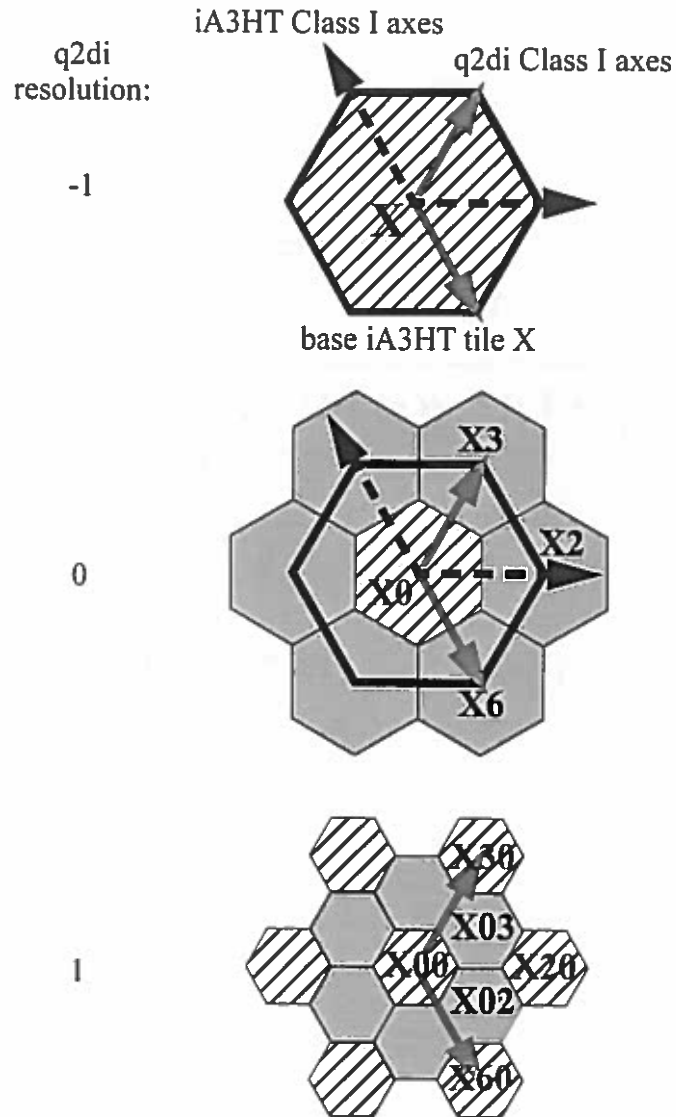


Figure 41. Relationship between base iA3HT tiles and the corresponding q2di quadrilateral.

The first case requiring adjustment is that iA3HT base cells 0 and 31 (corresponding to the unique single-cell pseudo-quadrilaterals 0 and 11) require additional sub-rotations to account for their unique orientation relative to the standard quadrilaterals. These rotations are indicated in the fourth column of Table 9.

We must also take into account the pentagonal cells which we have ignored up to this point. To accomplish this we check for cases where the generated address lies on a deleted sub-sequence of an A3PT tile. These cases occur when the cell has a prefix as indicated in the fifth column of Table 9. In these cases we must perform the sub-rotation indicated in the sixth column so that the appropriate existing portion of the pentagon replaces the interruption.

Finally, note that, as previously discussed, Class II q2di cells are addressed using the codes of the next finer resolution Class I q2di grid. Given a Class II q2di address $\{r, [q, (i, j)]\}$ first apply the above procedure for the address $\{r+1, [q, (i, j)]\}$. Since this address is, by definition, the Class I center cell of the desired Class II cell, the resulting iA3HT code will be the address of the desired Class II cell with an extra final digit of zero. Dropping this final zero digit from the code will give the correct code corresponding to the desired Class II resolution cell.

Conclusions

In this chapter we have specified two path addressing location coding systems for icosahedral aperture 3 hexagon DGGs. These systems share many of the advantages of traditional tree-based path addressing systems, and it is our hope that these systems will enable the use of hexagon DGGs in a broader range of applications that have traditionally relied on tree-based hierarchies. These systems have been implemented and are in use as the location coding scheme for a geospatial data visualization application commissioned by the Canadian Space Agency. Considerable work remains, however, to define efficient algorithms on these systems before their full potential can be exploited.

Table 9. iA3HT base cell look-up and adjustments.

Q2DI Quad q	A3HT Prefix	iA3HT Base Cell	Cell 00/31 Sub-Rotation	Interruption Prefix	Interruption Sub-rotation
1	X00	10	-	-	-
	X02	11	-	-	-
	X03	01	-	-	-
	X20	06	-	061	60° CW
	X30	00	0°	-	-
	X60	21	-	-	-
2	X00	06	-	-	-
	X02	12	-	-	-
	X03	02	-	-	-
	X20	07	-	071	60° CW
	X30	00	60° CCW	-	-
	X60	22	-	-	-
3	X00	07	-	-	-
	X02	13	-	-	-
	X03	03	-	-	-
	X20	08	-	081	60° CW
	X30	00	120° CCW	-	-
	X60	23	-	-	-

Table 9. iA3HT base cell look-up and adjustments (continued).

Q2DI Quad q	A3HT Prefix	iA3HT Base Cell	Cell 00/31 Sub-Rotation	Interruption Prefix	Interruption Sub-rotation
4	X00	08	-	-	-
	X02	14	-	-	-
	X03	04	-	-	-
	X20	09	-	091	60° CW
	X30	00	180° CCW	001	60° CCW
	X60	24	-	-	-
5	X00	09	-	-	-
	X02	15	-	-	-
	X03	05	-	-	-
	X20	10	-	101	60° CW
	X30	00	60° CW	-	-
	X60	25	-	-	-
6	X00	21	-	216	60° CW
	X02	26	-	-	-
	X03	16	-	-	-
	X20	22	-	-	-
	X30	06	-	-	-
	X60	31	60° CW	-	-
7	X00	22	-	226	60° CW
	X02	27	-	-	-
	X03	17	-	-	-
	X20	23	-	-	-
	X30	07	-	-	-
	X60	31	180° CCW	316	60° CCW

Table 9. iA3HT base cell look-up and adjustments (continued).

Q2DI Quad q	A3HT Prefix	iA3HT Base Cell	Cell 00/31 Sub-Rotation	Interruption Prefix	Interruption Sub-rotation
8	X00	23	-	236	60 ⁰ CW
	X02	28	-	-	-
	X03	18	-	-	-
	X20	24	-	-	-
	X30	08	-	-	-
	X60	31	120 ⁰ CCW	-	-
9	X00	24	-	246	60 ⁰ CW
	X02	29	-	-	-
	X03	19	-	-	-
	X20	25	-	-	-
	X30	09	-	-	-
	X60	31	60 ⁰ CCW	-	-
10	X00	25	-	256	60 ⁰ CW
	X02	30	-	-	-
	X03	20	-	-	-
	X20	21	-	-	-
	X30	10	-	-	-
	X60	31	0 ⁰	-	-

CHAPTER VI

Summary and Conclusions

In the current work we have attempted to develop the concept of DGGs as a new and promising paradigm for the structured representation of geospatial location. We begin here by reviewing the steps in that development.

Because DGGs do not fit neatly into the traditional classifications of spatial data structures, it was necessary to begin by taking a fresh look at the notion of structured geospatial data structures; that is, of data structures that have as their primary motivation the representation of geospatial location in as complete a form as possible. This led us necessarily to the development of an abstract data type for structured geospatial data structures that defined the boundaries of our explorations in the chapters that followed.

In Chapter II we defined the basic concepts associated with DGGs, and developed a system of classification that allowed us to identify the design criteria that must be evaluated in making comparisons and choices between DGGs alternatives. We then used this taxonomy to survey the primary DGGs approaches.

Our analysis in the first two chapters led us to conclude that, among the major DGGs design decisions, the choice of grid topology would appear to have the greatest potential impact on the performance of DGGs as data structures. We noted that the hexagon topology is considered the most promising approach by many users. But, because hexagon-based DGGs do not induce traditional tree hierarchies, they have remained problematic for, and consequently little-studied by, the data structures community. We recognized that in order to evaluate the implications of topology choice on data structure performance, we required a topology-independent implementation of the DGGs paradigm.

Based on our ADT definition we defined and implemented *nulib*, a uniform inter-

face to DGGs that at its core is independent of topology. As part of our implementation we generalized the notion of hierarchy to the notion of a spatial hierarchy, defining a set of hierarchical operators that apply not only to traditional trees but also to the regular, but non-tree, hierarchies induced by multi-resolution hexagon grids. We then demonstrated the suitability of nolib for designing and implementing real-world applications in a topology-independent fashion by using it to develop a discrete simulation architecture.

Armed with a platform for conducting empirical comparisons between DGG topologies, in Chapter IV we implemented what we believe to be the first spherical, multi-resolution, topology-independent discrete spatial simulation. We conducted initial experiments that indicated that the choice of grid topology does indeed effect the results of similar simulations.

In Chapter V we turned our attention to the primary remaining obstacle to the practical adoption of hexagon-based DGGs: the lack of efficient hierarchical location coding schemes for them. We discussed existing pyramid-based location codings for aperture 3 hexagon grids, such as the increasingly popular ISEA3H grid. We then introduced novel hierarchical systems for location coding vector and raster data in aperture 3 hexagon grids.

Discrete dynamic simulation on DGGs provides many avenues for further research. These include exploring the universe of possible interesting rule sets on spherical multi-resolution simulations for each specific topology in isolation, as well as attempting to clearly understand the relationship between similar rule sets defined on different topologies.

But in addition to providing a spatial substrate for discrete simulations, we have observed that DGGs can form the basis of vector, raster, bucket, and graph-based geospatial data structures. And with our introduction of a hierarchical location coding scheme for aperture 3 hexagon grids, we believe that hexagonal grids will begin to be used for these purposes. It would seem prudent, before the widespread adoption of particular DGG topologies for any of these important data structure classes, to thoroughly study the effect that the choice of DGG topology may have on the performance charac-

teristics of such data structures. We believe that the current work provides the theoretical foundation and practical tools necessary to conduct such studies, enabling both analytical and empirical comparisons between the DGGS topology alternatives.

BIBLIOGRAPHY

- Abu-Bakar, S., & Green, R.J. (1996). Detection of edges based on hexagonal pixel formats. *3rd International Conference on Signal Processing Proceedings (ICSP-96)*, Beijing, China, 1114-1117.
- Alborzi, H., & Samet, H. (2000, March). *Augmenting SAND with a spherical data model*. Paper presented at the First International Conference on Discrete Global Grids, Santa Barbara, CA.
- Bailey, H.P. (1956). Two grid systems that divide the entire surface of the Earth into quadrilaterals of equal area. *Transactions of the American Geophysical Union*, 37, 628-635.
- Baumgardner, J.R., & Fredrickson, P.O. (1985). Icosahedral discretization of the two-sphere. *SIAM Journal of Numerical Analysis*, 22(6), 1107-1114.
- Bell, S.B., Diaz, B.M., Holroyd, F., & Jackson, M.J. (1983). Spatially referenced methods of processing raster and vector data. *Image and Vision Computing*, 1(4), 211-220.
- Brooks, D.R. (1981). *Grid systems for Earth radiation budget experiment applications*, NASA Technical Memorandum 83233. (NTIS No. N82-14818)
- Burt, P.J. (1980). Tree and pyramid structures for coding hexagonally sampled binary images. *Computer Graphics and Image Processing*, 14, 71-280.
- Casati, R., & Varzi, A. (1996). The structure of spatial localization. *Philosophical Studies*, 82, 205-239.
- Chen, J., Zhao, X., & Li., Z. (2003). An algorithm for the generation of Voronoi diagrams on the sphere based on QTM. *Photogrammetric Engineering & Remote Sensing*, 69(1), 79-90.
- Christaller, W. (1966). *Central Places in Southern Germany*. Englewood Cliffs, NJ: Prentice Hall.
- Clarke, K.C. (2002). Criteria and measures for the comparison of global geocoding systems. In M.F. Goodchild & A.J. Kimerling (Eds.), *Discrete global grids: A web book*. Santa Barbara, CA: University of California. Retrieved June 1, 2005, from <http://www.ncgia.ucsb.edu/globalgrids-book/comparison>

- Conway, J.H., & Sloane, N.J.A. (1998). Coverings, lattices and quantizers. In *Sphere Packings, Lattices, and Groups* (pp. 56-62). New York: Springer-Verlag.
- Dacey, M.F. (1965). The geometry of central place theory. *Geografiska Annaler*, 47, 111-124.
- Dutton, G. (1984). Geodesic modelling of planetary relief. *Cartographica, Monograph 32-33*, 21, 188-207.
- Dutton, G. (1989). The fallacy of coordinates. *Multiple representations: Scientific report for the specialist meeting*. Santa Barbara, CA: National Center for Geographic Information and Analysis.
- Dutton, G. (1999). *A hierarchical coordinate system for geoprocessing and cartography*. Berlin, Germany: Springer-Verlag.
- Fekete, G., & Treinish, L. (1990). Sphere quadtrees: A new data structure to support the visualization of spherically distributed data. *SPIE, Extracting Meaning from Complex Data: Processing, Display, Interaction*, 1259, 242-25.
- Fredkin, E. (1960). Trie memory. *Communications of the ACM*, 3(9), 490-499.
- Frisch, U., Hasslacher, B., & Pomeau, Y. (1986). Lattice-gas automata for the Navier-Stokes equations. *Physics Review Letters*, 56, 1505-1508.
- Fuller, R.B. (1975). *Synergetics*. New York: MacMillan.
- Gardner, M. (1970). The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223, 120-123.
- Gibson, L., & Lucas, D. (1982). Spatial data processing using generalized balanced ternary. In *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing*. Las Vegas, NV: IEEE Computer Society. 566-571.
- Golay, J.E. (1969). Hexagonal parallel pattern transformations. *IEEE Transactions on Computers*, C-18(8), 733-739.
- Goodchild, M.F., & Yang, S. (1992). A hierarchical spatial data structure for global geographic information systems. *CVGIP: Graphical Models and Image Processing*, 54(1), 31-44.

- Gray, R. W. (1995). Exact transformation equations for Fuller's world map. *Cartographica*, 32(3), 17-25.
- Gregory, M. (1999). *Comparing inter-cell distance and cell wall midpoint criteria for discrete global grid systems*. Unpublished master's thesis, Oregon State University, Corvallis.
- Guttman, A. (1984). R-tree: A dynamic index structure for spatial searching. In *SIGMOD '84, Proceedings of the ACM SIGMOD International Conference on the Management of Data*, 47-57.
- Hastings, D.A., & Dunbar, P.K. (1998). Development and assessment of the global one-km base elevation digital elevation model (GLOBE). *ISPRS Archives*, 32(4), 218-21.
- Heikes, R., & Randall, D.A. (1995a). Numerical integration of the shallow-water equations on a twisted icosahedral grid. Part I: Basic design and results of tests. *Monthly Weather Review*, 123(6), 1862-1880.
- Heikes, R., & Randall, D.A. (1995b). Numerical integration of the shallow-water equations on a twisted icosahedral grid. Part II: A detailed description of the grid and an analysis of numerical accuracy. *Monthly Weather Review*, 123(6), 1881-1887.
- Kenner, H. (1976). *Geodesic math and how to use it*. Berkeley, CA: University of California Press.
- Kiester, R., & Sahr, K. (2000, March). *Unexpected and complex behavior of hierarchical, multiresolution cellular automata*. Paper presented at the First International Conference on Discrete Global Grids, Santa Barbara, CA.
- Kimerling, A.J., Sahr, K., White, D., & Song, L. (1999). Comparing geometrical properties of global grids. *Cartography and Geographic Information Science*, 26(4), 271-287.
- Kurihara, Y. (1965). Numerical integration of the primitive equations on a spherical grid. *Monthly Weather Review*, 93, 399-415.
- Lawler, E., Lenstra, J., Rinnooy Kan, A., & Shmoys, D. (1985). *The traveling salesman problem*. New York: John Wiley.
- Luczak, E., & Rosenfeld, A. (1976). Distance on a hexagonal grid. *IEEE Transactions on Computers*, C-25(5), 532-533.

- Lukatela, H. (2002). A seamless global terrain model in the Hipparchus system. In M.F. Goodchild & A.J. Kimerling (Eds.), *Discrete global grids: A web book*. Santa Barbara, CA: University of California. Retrieved June 1, 2005, from <http://www.ncgia.ucsb.edu/globalgrids-book/terra>
- Nievergelt, J. (1989). 7 +/- 2 criteria for assessing and comparing spatial data structures. In A. Buchman, O. Gunther, T. Smith, & Y. Wang (Eds.), *Design and implementation of large spatial databases: Proceedings of the First Symposium SSD '89* (pp. 3-27). Berlin: Springer-Verlag.
- Noronha, V. (2000, March). *Design issues in the Go2 grid system*. Paper presented at the First International Conference on Discrete Global Grids, Santa Barbara, CA.
- Olsen, A.R., Stevens, D.L., & White, D. (1998). Application of global grids in environmental sampling. In S. Weisberg (ed.), *Computing science and statistics (volume 30): Proceedings of the 30th Symposium on the interface, computing science and statistics* (pp. 279-284). Fairfax Station, VA.: Interface Foundation of North America.
- Otoo, E.J., & Zhu, H. (1993). Indexing on spherical surfaces using semi-quadcodes. In D.J. Abel & B.C. Ooi (Eds.), *Advances in spatial databases: Proceedings of the Third International Symposium on advances in spatial databases* (pp. 510-529). Singapore: Springer-Verlag.
- Paul, M.K. (1973). On computation of equal area blocks. *Bulletin Geodésique*, 107, 73-84.
- Rigaux, P., Scholl, M., & Voisard, A. (2002). *Spatial databases with application to GIS*. New York: Morgan Kaufmann.
- Ringler, T. D., & Heikes, R.P. (1999). *Modeling the atmospheric general circulation using a spherical geodesic grid: A technical report on a new class of dynamical cores*. Colorado State University.
- Rothman, D.H., & Zaleski, S. (1997). *Lattice-gas cellular automata: Simple models of complex hydrodynamics*. Cambridge: Cambridge University Press.
- Sadourny, R., Arakawa, A., & Mintz, Y. (1968). Integration of the nondivergent barotropic vorticity equation with an icosahedral-hexagonal grid for the sphere. *Monthly Weather Review*, 96(6), 351-356.
- Saff, E.B., & Kuijlaars, A. (1997). Distributing many points on a sphere. *Mathematical Intelligencer*, 19(1), 5-11.

- Sahr, K., & White, D. (1998). Discrete global grid systems. In S. Weisberg (ed.), *Computing science and statistics (volume 30): Proceedings of the 30th Symposium on the interface, computing science and statistics* (pp. 269-278). Fairfax Station, VA.: Interface Foundation of North America.
- Sahr, K. (2002). *DGGRID: User documentation for discrete global grid software*. Retrieved June 1, 2005, from <http://www.sou.edu/cs/sahr/dgg/dggrid/docs/dggridoc31.pdf>
- Sahr, K., White, D., & Kimerling, A.J. (2003). Geodesic discrete global grid systems. *Cartography and Geographic Information Science*, 30(2), 121-134.
- Sahr, K., Peterson, P., & Lutterodt, L. (2003) *The aperture 3 hexagon tree*. Unpublished manuscript.
- Samet, H. (1989). *The design and analysis of spatial data structures*. Menlo Park, CA: Addison-Wesley.
- Samet, H. (1990). *Applications of spatial data structures: Computer graphics, image processing and GIS*. Menlo Park, CA: Addison-Wesley.
- Shekhar, S., Chawla, S., Ravada, S., Fetterer, A., Liu, X., & Lu, C. (1999). Spatial databases - accomplishments and research needs. *IEEE Transactions on Knowledge and Data Engineering*, 11(10), 45-55.
- Shekhar, S., & Chawla, S. (2003). *Spatial databases: A tour*. Upper Saddle River, NJ: Prentice-Hall.
- Shumway, R.H. (2000). *Time series analysis and its applications*. New York: Springer Verlag.
- Snyder, J. P. (1992). An equal-area map projection for polyhedral globes. *Cartographica*, 29(1), 10-21.
- Snyder, W., Qi, H., & Sander, W. (1999). A coordinate system for hexagonal pixels, *Proceedings of SPIE, the international society for optical engineering*, 3661, 716-727.

- Song, L., Kimerling, A.J., & Sahr, K. (2002). Developing an equal area global grid by small circle subdivision. In M.F. Goodchild & A.J. Kimerling (Eds.), *Discrete global grids: A web book*. Santa Barbara, CA: University of California. Retrieved June 1, 2005, from <http://www.ncgia.ucsb.edu/globalgrids-book/song-kimmerling-sahr>
- Stefanakis, E., & Kavouras, M. (1995). On the determination of the optimum path in space. In *Proceedings of the 2nd International Conference on Spatial Information Theory (COSIT '95)* (pp. 241-257). New York: Springer.
- Thurn, J. (1997). A PV-based shallow-water model on a hexagonal-icosahedral grid. *Monthly Weather Review*, 125, 2328-2347.
- Tobler, W.R., & Chen, Z. (1986). A quadtree for global information storage. *Geographical Analysis*, 18(4), 360-71.
- van Roessel, J. (1988). Conversion of cartesian coordinates from and to generalized balanced ternary addresses. *Photogrammetric Engineering and Remote Sensing*, 54(11), 1565-1570.
- Verbrugge, C. (1997). *Hex grids*. Unpublished manuscript, McGill University, Montreal, Quebec, Canada.
- von Neumann, J. (1966). *The Theory of Self-Reproducing Automata*. University of Illinois Press.
- White, D., Kimerling, A. J., & Overton, W. S. (1992). Cartographic and geometric components of a global sampling design for environmental monitoring. *Cartography and Geographic Information Systems*, 19(1), 5-22.
- White, D., Kimerling, A. J., Sahr, K. & Song, L. (1998). Comparing area and shape distortion on polyhedral-based recursive partitions of the sphere. *International Journal of Geographical Information Science*, 12, 805-827.
- White, D. (2000). Global grids from recursive diamond subdivisions of the surface of an octahedron or icosahedron. *Environmental Monitoring and Assessment*, 64(1): 93-103.
- Wickman, F. E., Elvers, E. & Edvarson, K. (1974). A system of domains for global sampling problems. *Geografiska Annaler*, 56(3/4), 201-212.
- Williamson, D. L. (1968). Integration of the barotropic vorticity equation on a spherical geodesic grid. *Tellus*, 20(4), 642-653.

Wuthrich, C.A., & Stucki, P. (1991). An algorithmic comparison between square- and hexagonal-based grids. *CVGIP: Graphical Models and Image Processing*, 53(4), 324-339.

D

D

D

D

D

D

D

D

D

D

D

