COMPUTER CARTOGRAPHY FOR GIS: AN OBJECT-ORIENTED VIEW ON THE DISPLAY TRANSFORMATION

ANDREW U. FRANK and MAX J. EGENHOFER

Department of Surveying Engineering and National Center for Geographic Information and Analysis, University of Maine, Orono, ME 04469, U.S.A.

Abstract—Geographic Information Systems (GISs) are widely used tools for the collection, management, and display—or visualization—of many types of data that describe space. Visualization of spatial data has been the domain of expertise of cartographers and elaborate recommendations for best rendering of spatial data exist. Unfortunately, this body of knowledge is not cast yet into a formalization and thus is not accessible immediately for programming GIS software.

A particular problem is the description of the rendering parameters for complex spatial objects. This paper presents a method for describing the set of individual geometric objects parts to which different rendering parameters can be assigned. The geometric data model uses the concepts of boundary and interior, and their specializations returning objects of particular dimensions. It is applicable equally to both raster and vector data, and, therefore, a contribution to the integration of vector and taster GIS. The rendering parameters are based upon Bertin's "visual variables." Abbreviated class definitions in C++ are included as a method to describe formally the concepts treated.

Key Words: Object-oriented programming, C++, Computer cartography, Visual variables, Topology, Boundary, Interior.

INTRODUCTION

A Geographic Information System (GIS) is a tool for the collection, management, and display of spatial information (Burrough, 1986). Many organizations adopt GISs to treat data that describe space, spatial objects, or human activities related to location in space. Spatial data range from information about the boundaries and ownership of land parcels (NRC, 1980) over regional land use to climate data on a global scale (Mounsey and Tomlinson, 1988).

A critical aspect of a GIS is the visual communication of results (Egenhofer, 1990). Some GISs offer interactive query languages—most are extensions of SQL (Ingram and Phillips, 1987; Roussopoulos, Faloutsos, and Sellis, 1988; Herring, Larsen, and Shivakumar, 1988; Egenhofer, 1991a)—which increase the users' abilities to *select* data based on spatial criteria; however, they support only tabular representations of query results and do not sufficiently address the *rendering* of the data selected (Egenhofer, 1991b). Spatial data presented in nonspatial, for example tabular, format are difficult to read and most of the information cannot be "seen" without graphic rendering (Tufte, 1983, 1990).

The topic of this paper is the translation of attribute data into graphical form. Cartography has developed a large collection of elaborate rules for displaying spatial information. Unfortunately, most of these rules are informal recommendations that require "common sense" to be interpreted (Imhof, 1972; Robinson and others, 1984). The users' expectations of flexibility in the graphical presentation of the results cannot be met by the state of the art in computer cartography. Although some special problems of computer cartography, primarily placement of names (Freeman and Ahn, 1987), have been studied, currently, one cannot automatically construct maps and diagrams on demand so that they optimally communicate a given set of data.

This paper follows the tradition of understanding cartography as a transformation (Tobler, 1979; Robinson and others, 1984). Cartographers have studied extensively the transformations between point sets (map projections) and the transformations from a cartographic line to another (line generalization). Here, we concentrate on the specific transformation of spatially related attribute data into a graphical representation, namely the values and parameters necessary to determine the graphical rendering for geometric elements. The paper develops the chain of transformations from the geometric and attribute data stored in a database to the display. In this process, some of the geometric attributes are selected to determine the position and form of the object rendered. Other attributes are selected to determine the color, pattern, and line width of the presentation of the object. A variation of Bertin's (1983) "visual variables" is used to describe the visual parameters of each component of the geometric object. This process is applicable to both vector and raster data, thus making a contribution to their desired integration in GIS (Ehlers, Edwards, and Bedard, 1989). Obviously, the details of the transformation depend on the geometric data model used (Egenhofer and Herring, 1991; Frank, 1992). For raster data, only one visual parameter descriptor is necessary, whereas for vector data, modeled as n-dimensional simplicial complexes (Frank and Kuhn, 1986; Egenhofer, Frank, and Jackson, 1989) or cell complexes (Herring, 1987), $2n + 1$ descriptors are necessary to describe completely all topologically distinct parts that can be rendered differently (Egenhofer, 1988).

This paper includes the simplified class definitions, written in C++ (Stroustrup, 1986), to describe formally the concepts involved. It exploits the object-oriented features of this language, particularly inheritance and encapsulation. Such code in a machine-checkable specification language is an invaluable contribution to the development of a high-level software design. Pseudo C++, used as a high-level specification and design language, has the advantage that no transition occurs when moving to coding. It demonstrates that an object-oriented view can be used to model a sequence of transformations, traditionally thought of as a set of procedures.

The remainder of this paper focuses on the transformations that are necessary for rendering geometric objects as a map display. After a brief review of geometric data models and object-oriented software design methods, the investigations focus on the separation of the graphical rendering process into (1) the attributes that describe the objects and (2) the visual variables to render them. Formal models for spatial objects and their properties are described and then the transformation process is laid out. Following an object-oriented design, this transformation is a stepwise transformation from a data object, gradually becoming more and more display-oriented. Finally, the number of descriptors necessary for multidimensional objects is described.

DATA MODEL FOR GEOMETRIC OBJECTS

Spatial concepts are comprised of ideas, notions, and relations between spatial objects, which humans use to organize and structure their perception of reality. Spatial concepts differ depending on the task at hand, the circumstances, and the experience of the persons. They are described only informally or based on such concepts as infinite sequences, so that they cannot be implemented directly because of fundamental restrictions of computer systems such as their finiteness.

A *data model* is a set of objects with a formal definition of the appropriate operations and integrity rules formally defined. The term, primarily used in the database community (Ullman, 1982; Date, 1986), is fully applicable to spatial and geometric information (Egenhofer and Herring, 1991; Frank, 1992). A data model is suitable for computer implementation and, therefore, must be discrete. A data model is similar to an algebra, as it consists of a set of objects, a set of operations that can be applied to these objects, and rules that define the results of the application of operations to objects. A *spatial data model* is a comprehensive set of conceptual tools to be used to structure spatial data, including the description of data and appropriate operations. Spatial data models are defined and constructed formally such that they can be implemented (i.e. discrete).

A data model is implemented by selecting a *data structure*, which provides the operations defined for the data model, and mapping them onto the code specific for the data structure. *Spatial data structures* are low-level descriptions of storage structures and the pertinent operations, with details of how to achieve the desired effects. Although they provide a specific function, that is fulfill the conditions of an operation, they also are fixed in terms of performance and storage utilization. Numerous spatial data structures have been proposed for spatial data in a GIS (Samet, 1989b).

The two major spatial data models used in GIS are the raster data model and the vector data model. The *raster model* is based on a regular raster that divides space into regularly shaped and sized cells. It records for each cell attribute values, which describe the nongeometric properties at the corresponding location. The typical operations on the raster data model combine the attribute values for one raster cell, using the values for different properties, to compute a new data value for the same cell. This is a computational form of spatial overlay (Chan and White, 1987; Tomlin, 1990), which directly relates to the manual practice used by planners in the past (Steinitz, Parker, and Jorden, 1976). A multitude of spatial data structures can be used to implement the raster data model as surveyed by Samet (1989a). The other widely used spatial data model is the *vector* or *topological data model*. It is based on a subdivision of space in irregularly shaped regions with their boundaries formed by lines that link points. This vector data model uses the concept of topology (Alexandroff, 1961) and includes operations to locate the boundary and the interior of a given object. A standard implementation uses tables, but there are other implementations that provide the same functionality (Herring, 1987; Güting, 1988).

OBJECT-ORIENTED SOFTWARE DESIGN

Currently, an object-oriented approach is being promoted as the most appropriate method for modeling complex situations that are concerned with real-world phenomena and, therefore, is applicable to GIS (Egenhofer and Frank, 1987, 1989; Worboys, Hearnshaw, and Maguire, 1990). A definition of object-orientation is that an entity of whatever complexity and structure can be represented by exactly one object (Dittrich, 1986; Zdonik and Maier, 1990). No artificial decomposition into simpler parts should be necessary resulting from technical restrictions, for example normalization rules (Codd, 1972). Object-oriented concepts are more flexible and powerful than the traditional models such as the relational (Codd, 1970) or the entity-relationship data model (Chen, 1976). They can both be shown to be subsets of an object-oriented data model.

Object-oriented abstraction mechanisms

Data abstraction is a method of modeling data. Object-oriented design uses three major abstraction mechanisms (Brodie, Mylopoulos, and Schmidt, 1984): (1) classification, (2) aggregation, and (3) generalization. Previous methods, such as the relational model, lacked some and had to simulate them with other constructions (Meyer, 1988). *Classification* can be expressed as the mapping of several objects (instances) onto a common class. In the object-oriented approach, every object is an instance of a class. A class characterizes the behavior of its instances by describing the operators that can manipulate those objects. *Generalization* provides for the grouping of classes of objects, which have some operations in common, into a more general superclass. Instances of objects of the subclass and superclass are related by an *isa*-relation, because each instance of a subclass is also an instance of a superclass. *Inheritance* describes in a generalization hierarchy the behavior of instances of a subclass in terms of its superclass. *Single inheritance* (Goldberg and Robson, 1983) implies that each subclass belongs at most to a single superclass, whereas *multiple inheritance* (Cardelli, 1984) permits classes with several distinct superclasses. *Aggregation* allows for the combination of several objects to a semantically higher-level object where each part has its own functionality. This abstraction usually is referred to as the *partof*-relation, because each component object is part of the aggregate or composite object.

Object-oriented programming languages

Many of the object-oriented concepts were initially proposed in the programming language Simula (Dahl and Nygaard, 1966). Although several other object-oriented programming languages have been developed since (Goldberg and Robson, 1983; Meyer, 1988), only C++ (Stroustrup, 1986) has become a widely used programming language that includes all abstraction methods in a single language. C++ compilers exist for a wide variety of computer systems and produce fast and efficient code. In this paper, C++ is used as a high-level specification language. It allows us to write formal descriptions of the data objects and the applicable operations. Although C++ is not ideally suited for this task (Khoshafian and Abnous, 1990), the advantages of checking specifications for type consistency outweigh the disadvantages of some of the limitations, particularly the "define before use" rule. Typical specification languages, such as Larch (Guttag, Horning, and Wing, 1985), differ too strongly from any implementation language so that another translation step from the specification to the implementation becomes necessary; however, the primary goal in this paper is to describe an overall structure of objects and operations, and the relations among them. Most details of an actual implementation are only sketched; however, the benefits of having the compiler check that the pieces are complete and fit together warrant the effort.

SOURCE AND TARGET DOMAIN OF RENDERING

Graphical rendering is understood as a transformation process, mapping from the abstract, internal representation of objects in a computer format onto a graphical rendering, which humans understand. In the rendering transformation, a number of problems must be resolved such as the selection of the objects to be rendered and the appropriate transformation of the metric information from world coordinates to map coordinates (Clarke, 1990). These problems of cartographic generalization and map projections are excluded here in order to concentrate on the step of graphically rendering objects with an irregular geometry that is representative of some aspects of their form.

The transformation concept leads to the question, "What are the source and target domains of the graphical representation?" The *source domain* is a selected set of objects that should be rendered. In each object some geometric data are selected to control the geometric form of the object; it must be expected that future GISs may contain objects with multiple geometric representations (Buttenfield, 1989), for example a road is represented internally as a region or a center line and one of the two is selected as the geometry of the object controlling rendering. Then some of the other attribute values are selected to control other aspects of the graphical appearance of the object, for example, the building type or the nationality of the owner to provide a thematic map. The *target domain* is the set of visual variables, that is the graphical components humans can differentiate, primarily color and line width.

Attribute values

The values that describe the properties of spatial objects in GISs are of many different types, including land use classification, land value, forest stand type, soil types, etc. Stevens (1946) classified attribute values into four groups:

- nominal, for example the names of persons or land use classes, such as residential, industrial, and rural;
- ordinal, for example classifications such as unsuitable, suitable, and very suitable;

- interval, for example temperature in Fahrenheit; and
- ratio, for example the value in dollars per square foot.

This grouping differentiates properties of the values that determine the types of statistical analysis applicable and influences the selection of methods to translate a value into a graphical form. Mackinlay (1986) uses a slightly simplified classification combining interval and ratio data to guide the selection of the appropriate graphical representation. Bertin (1977) indicates that some other properties of data values are important in cartographic rendering, for example, if a value relates to the total area or if it is an average per unit area.

The structure of the source domain of the graphical representation is formalized by user-defined classes, which are specializations of the superclass **Attribute-Value**. For example, the classes of forest stands, land use, and land value may be defined as records consisting of enumerated types and floating point numbers, respectively. Each class is a subclass of the class **AttributeValue** from which it inherits the corresponding operations for attributes of type nominal, ordinal, interval, and ratio.

```
class AttributeValue {
        enum kind {nominal, ordinal, interval, ratio} k;
};

class StandType : AttributeValue {
        enum stand { birch, fir, spruce, beech, oak } s;
};

class LandUse : AttributeValue {
        enum landuses {residential, agricultural, industrial} l;
};

class LandValue : AttributeValue {
        float landprice;
};
```

A theme is an attribute of a data collection and indicates what type of data values to expect:

```
enum theme {
    Standtype,
    Landuse,
    Landvalue
};
```

Visual variables

Bertin (1983) distinguishes eight visual variables. The following list of parameters is adapted from his initial description (Bertin, 1977):

- The position in two dimensions on the display surface;
- the size of the symbol;
- color with the three variables luminance, hue, and saturation;
- orientation;
- pattern; and
- form of the symbol.

Generally, any of these variables can be used to express an aspect of the properties of the data set. Unlike business graphics where the position on the display surface can be used to convey a variable (Mackinlay, 1986), cartographic renderings automatically bind the position on the display to the location of the mapped object; therefore, this variable is not available for communicating attribute values (Schlichtmann, 1984).

The structure of the target domain of the representation is formalized by the classes **Color, SymbolForm, Orientation**, etc., which are specializations of the superclass **VisualVariable**. The following class definitions formalize this structure:

```
class VisualVariable {};


class Color : VisualVariable {
            float hue;
            float lightness;
            float saturation;
};


class SymbolForm : VisualVariable {
           enum symbolform {rectangular, circle, square, cross} s;
};


class Orientation : VisualVariable {
           float o;  //an angular value from 0 to 360
};


class Size : VisualVariable {
           float size;  //scale factor
};


class Pattern : VisualVariable {
           enum patternTypes {blank, hatched, crosshatched, grey, black} p;
};
```

Newer displays and recent work in computer graphics indicate another approach, that is mapping the variables onto a 3-D surface, draped with different textured and patterned coverages (Robertson, 1988). This exploits the special human ability to discern subtle changes and interpret form, but also adds new visual variables to the list such as specula and reflectance.

All the visual variables for a graphical patch are collected in *patch properties*.

```
class PatchProp {
           Color c;
           SymbolForm sf;
           Orientation o;
           Size s;
           Pattern p;
};
```

Graphical communication

Graphical rendering of data is based on a mapping from data values onto graphical variables, which the human observer can discern and interpret. In a map legend, the meaning of the graphical properties is given in terms of the data values in order to allow for this interpretation. Humans can distinguish a number of different graphical properties of a "patch" on a display surface. In order to communicate properly, it must be assured that at least the scales of the data values and the apparent scales of the graphical variable are the same, that is that the graphical representations express the same relations the data have. For example, nominal data, such as land use classes, can be well represented by different symbols, because both land use classes and symbols are on a nominal scale. The representation of land value, a value on a ratio scale, requires a visual variable that allows for a comparison of ratios such as a gray scale.

GEOMETRY AND SPATIAL OBJECTS

The geometry selected to control rendering primarily determines location and extent of the graphical object. For example, the location and the extent of a stand of forest are described by a boundary polygon, a list of the raster cells covered, or any other suitable

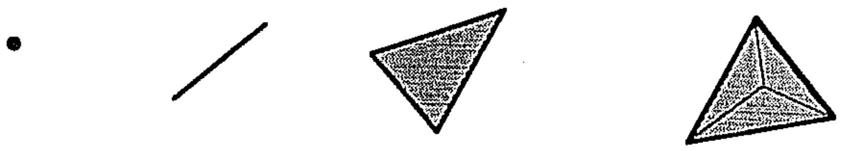


Figure 1. 0-, 1-, 2-, and 3-simplex.

method for describing the geometry of the stand. Its properties are the type of forest stand, the age of the stand, and the land value, all expressed as values of the appropriate types. This section discusses the geometric descriptions encoded in the class **GeometricObject** and then forms the class **SpatialObject**. For both exist abstract objects that provide some operations, which will be described later.

Raster data are modeled as a grid of cells where each cell has a position and a set of attributable values (Dorenbeck and Egenhofer, 1991). It is a mapping from position and theme (type of attribute) onto an attribute value. The type of the attribute and the values it can take depend on the theme. For example, common values for land use are "residential" or "forest," and with the map overlay operations one can construct new themes from existing ones (Tomlin, 1990).

```
        int x, y;
};


class GeometricObject {};


class RasterSquare : GeometricObject {
        position p;
        extend int;
};


class SpatialObject {
};


class RasterCell : SpatialObject {
};
```

Different approaches can be used to construct a vector-oriented data model based on topology. Recently, combinatorial topology was proposed to be exploited to model spatial data in GIS (Frank and Kuhn, 1986; Herring, 1987; Egenhofer, Frank, and Jackson, 1989). This was a further development of the classical use of topology in GIS (Corbett, 1979). Data models using this mathematical structure have been proposed both for two-dimensional (Egenhofer, 1987) and three-dimensional (Carlson, 1987) geometry. Their implementation demonstrated the simplicity of using a straight mathematical theory (Jackson, 1989). This section gives a brief summary of simplex theory and simplicial complexes and describes the operations boundary and interior, which are relevant to the view of spatial objects and their parts.

Simplex

Objects are classified according to their dimension. For each dimension, a minimal object exists, termed *simplex* (Munkres, 1966; Spanier, 1966).

- A point, the minimal object in a 0-dimensional space, is a 0-simplex;
- an edge is a 1-simplex;
- a triangle is a 2-simplex;
- a tetrahedron is a 3-simplex, etc. (Fig. 1).

This model is applicable for any n-dimensional space with an n-simplex being the minimal object. Any n-simplex contains $n + 1$ simplices of dimension $n - 1$ that are geometrically independent. For example, a triangle, a 2-simplex, is bounded by three 1-simplices. The 1-simplices are independent geometrically if there is no pair of parallel edges and no edge is of length 0. A *face* of a simplex S is any simplex that is contained in S. A node in the edge of a bounding triangle of a tetrahedron is a face as well as a bounding triangle. Each simplex has a dimension. The 0-simplex, as a special situation of the general simplex, has also a position attribute. The 1-simplex contains two 0-simplices for start and end, and the 2-simplex contains the three bounding lines (1-simplices).

```
class Simplex {
        int dimension};


class Simplex0 : Simplex {
        position n;
};


class Simplex1 : Simplex {
        Simplex0 start, end;
};


class Simplex2 : Simplex {
        Simplex1 first, second, third;
};
```

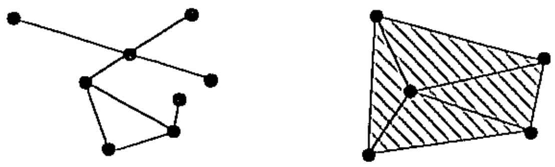


Figure 2. Examples of simplicial complexes.

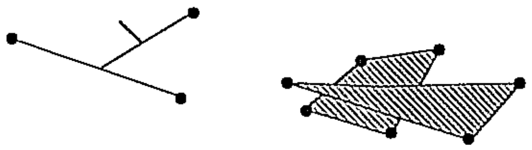


Figure 3. Examples of configurations that are not simplicial complexes.

Simplicial complex

A *simplicial complex* is a (finite) collection of simplices and their faces. If the intersection between two simplices of this collection is not empty then the intersection is a simplex that is a face of both simplices. For example, the configurations of Figure 2 are complexes, whereas Figure 3 shows three configurations that are not simplicial complexes. Simplicial complexes hold some favorable operations, such as boundary and interior. The set-theoretic *boundary* determines all bounding simplices of a complex (Fig. 4). The complementary operation to boundary is *interior*, which determines the set of all simplices that are not part of the boundary (Fig. 5).

```
class Complex : GeometricObject {
   public :
      void complex boundary() {
      // determines the bounding faces
      };
      void complex interior() {
      // determines the interior faces
      };
};

class Complex0 : Complex {
      \\ set of 0-simplices
};

class Complex1 : Complex {
      \\ set of 1-simplices
};

class Complex2 : Complex {
      \\ set of 2-simplices
};
```

Generalization to cells and cell complexes

The methods as described for simplices and simplicial complexes can be generalized to arbitrarily shaped *cells* (Fig. 6A) and *cell complexes* built from such (Fig. 6B). Formalization and programming of simplices and simplicial complexes is easier as they are of fixed topological structure, whereas cells may have a differing number of faces in their boundary. For this reason, the remainder of the paper assumes simplices, but the results are equally valid for the general situation of cells.

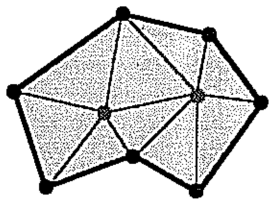


Figure 4. Bounding faces of simplicial complex.

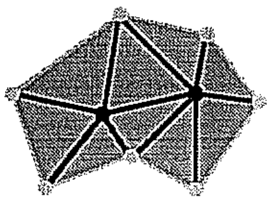


Figure 5. Interior faces of simplicial complex.

NUMBER OF PATCH DESCRIPTORS NECESSARY

A single patch descriptor is required if raster data are rendered. Most current systems use only a single visual variable, usually hue, maybe gray scale or gray scales simulated with patterns. It is clear that several of the visual variables can be differed at the same time and thus communicate multiple variables at the same time (Robertson, 1988). Yet, a single patch descriptor is sufficient.

For data modeled in a topological data model, the situation is more complex (Egenhofer, 1989):

- A point has ideally no dimension, but is represented graphically by a single patch and thus a single patch descriptor is sufficient. This is

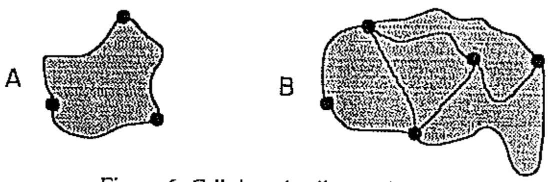


Figure 6. Cell A and cell complex B.

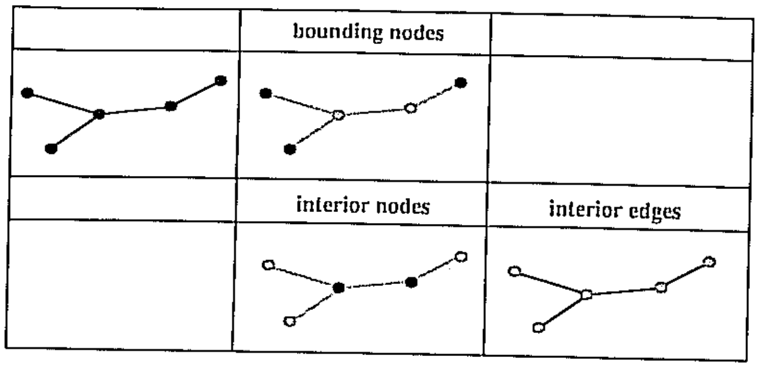


Figure 7. Bounding and interior faces of line.

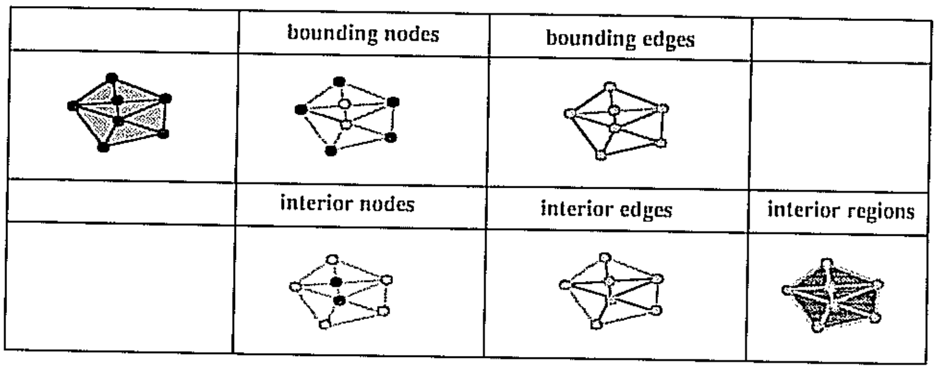


Figure 8. Bounding and interior faces of region.

essentially the situation for a raster cell, which is modeled as an extended point, arranged in a regular grid.

- A line is an aggregate of 1-simplices, each forming a single point-to-point link. The graphical representation consists of the line patch and patches for the bounding and interior nodes. In order to cover the most general situation, the rendering of the interior and bounding nodes must be directed by different descriptors. For example, a line may be drawn with the interior edges in blue, the bounding nodes in black, and the interior nodes in red (Fig. 7).
- A region is an aggregate of 2-simplices (triangles). Each subregion is bounded by edges that are bounded by nodes. An edge is in the boundary of an areal object if the edge bounds only one subregion which belongs to the object (Fig. 8).

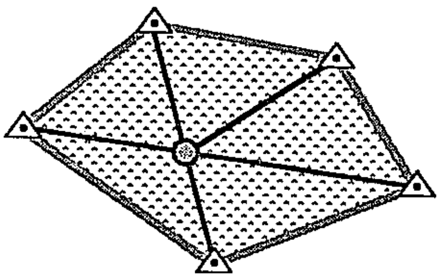


Figure 9. Assigning different visual variables to five parts of region.

The graphical representation of a region is specified fully by assigning a patch descriptor to each part of the region. The following parts of a region can be assigned to an individual set of graphical variables (Fig. 9):

- the area, that is the interior 2-dimensional faces of the region,
- the bounding edges,
- the interior edges,
- the nodes in the boundary, and
- the nodes in the interior.

This definition can be generalized so that it is independent of the particular dimension of the objects. The object parts of spatial objects of dimension n can be described by $2n+1$ definitions. In terms of simplicial complexes, the following parts can be assigned with patch descriptors:

- The interior of the object,
- all simplices of dimension $(n-1)\ldots 0$ that are part of the boundary of the complex, and
- all simplices of dimension $(n-1)\ldots 0$ that belong to the complex, but are not part of the boundary.

THE DISPLAY PROCESS

The display process is started by calling the operation **draw** for a **SpatialObject**. The following shows the individual steps of this process for a raster object. Corresponding operations are valid for the display of vector data.

```
main{}
{
  rasterDataSet forestData ;
  translationTable tt ;
  forestData.draw (tt);
};
```

The necessary setup operations, which control the transformation in a global sense, are assumed to be performed ahead of time. The result of the display process is the complete transformation from the spatial object to a displayable object that can be rendered by the basic graphics software.

Step 1: looping over all spatial objects in a data set

In order to render a collection of spatial objects, for example a **RasterDataSet**, each individual spatial object in the collection must be rendered. In a spatial data collection, the selection of the objects to be displayed may be based on a single theme or a small collection of themes.

Step 2: transforming an attribute value into a patch descriptor

The attribute value of a **SpatialObject** must be translated into a set of visual variables. It is necessary to set up a translation table that maps each attribute value onto a patch descriptor. The translation is potentially more complex than a mapping from a single data value onto a nominal scale onto a pattern type and can involve complex attribute values, constructed from several base values and mappings to multiple visual variables. There also may be "style rules," which assign default values to visual variables.

```
class TranslationTable {
   public :
      PatchProp translate( AttributeValue av );
};
```

Step 3: drawing a single spatial object

The command **draw** applied to a **SpatialObject** is executed by calling **draw** on the geometric part of the object with the **PatchDescriptor** as an argument (*shown overleaf*).

```
class RasterDataSet {
         rasterCell getCell( point p, theme t );
   public :
         void
         draw( theme t, TranslationTable tt ) {
            point p;
            rasterCell c;
            // for each position p do
            // (this is pseudo code, which is more intuitive than actual C++)
              {c = getCell( p, t );
               c.draw( tt );
              };
         };

};
```

```
class SpatialObject {
   public :
      virtual GeometricObject getGeometry ();
      virtual AttributeValue  getAttribute ();
      void
      draw( TranslationTable tt ) {
                 GeometricObject g;
                 g = getGeometry ();
                 g.draw( tt.translate( getAttribute() ) );
            };
      };
```

Step 4: transforming geometric object by view transformation

The **DisplayObject** contains the same type of geometry, suitably transformed from the object geometry expressed in world coordinates into the viewport coordinates; this transformation is well understood (Foley and others, 1990) and operates on the coordinate values. There is also an issue of reducing the number of points, for example in a line (Douglas and Peucker, 1973), if there are more points than can be shown on the display.

```
class GeometricObject {
   public :
     void draw( PatchProp p ) {
        // transforms the GeometricObject into a DisplayObject and draws it
     };
};
```

Step 5: rendering display object

The result of the previous step is a description of a graphical object that can be displayed by the regular display software. The **draw** command applied is executed by drawing the individual point symbols, lines, and areas, each with the appropriate patch descriptor.

```
class DisplayObject {
   public :
     virtual void draw( PatchProp p) = 0;
};
```

CONCLUSIONS

Spatially related attribute data are transformed for cartographic rendering. Their transformations must follow rules to communicate an intended meaning. The traditional cartographic transformational viewpoint (Tobler, 1979; Robinson and others, 1984) is used to define transformations of attribute data into graphical output. The attribute data are described in terms of Stevens' (1946) "level of measurement" and translated into Bertin's (1983) "visual variables" of the equivalent characteristic. Each area in a map must be characterized by a number of properties for color (usually three values), symbol form, and size. Each of these can be used to communicate one attribute value. If applied to a raster model of a map, each pixel is described by one such set of property values, termed a *patch descriptor*. If one uses other geometric data models (Egenhofer and Herring, 1991; Frank, 1992), several patch descriptors become necessary. This paper used a geometric data model based on combinatorial topology. Each map object is modeled as a set of cells of dimension 0, 1, or 2 with the appropriate boundaries. For example, the graphical rendering of an areal object, modeled as a simplicial complex, is described by the rendering of boundary and interior points; boundary and interior lines; and the interior area. The complete set of necessary patch descriptors for all objects in this geometric data model was developed.

This paper described only the most general situation, all the visual variables that must be determined in order to render a display object, and it did not touch on the problem of how these are determined from the attribute values of the spatial object. Clearly not all these visual variables can be used to differentiate one object from another. For example, it is not recommended to differentiate two types of land use classes, say residential and industrial, just by the size of the symbols used for showing the interior nodes—this would be an insufficient clue for a human map reader; however, it is necessary to state how these interior nodes are shown—or to state that they should not be shown. Current systems have many of these selections fixed or selected by default.

An important problem remains, namely how to select the visual variables to best represent the attribute values and thus the concepts people associate with them. There are some studies that show which visual variables are more forcefully or more accurately communicating some notions. Cleveland and McGill (1984) have studied the question for quantitative data but their findings need extension to the other situations.

The ultimate problem is the accurate rendering of human concepts of space and spatial objects. The concept of Euclidean space, applicable to the geometry on our displays, seems to be suitable also to large-scale, geographic spaces; however, people have different ways to conceptualize geographic (large) spaces, not all of them based on Euclidean concepts (Mark and others, 1989). The use of metaphors, which is the base for the mapping of attribute values to visual variables, is a powerful tool (Lakoff, 1987; Lakoff and Johnson, 1980; Kuhn and Frank, 1991), but not yet sufficiently understood.

Many difficult problems need solutions before a fully automatic translation of spatial data in a GIS to an understandable graphical presentation is possible. An approach is advocated that incrementally assists the human user, warns if inappropriate choices are made, and suggests possible solutions. The software engineering tools help us to build such system from the ground up, from the individual components to the overall processes.

Acknowledgments—Funding from Intergraph Corporation, Digital Equipment Corporation under TP-765536, and NSF for the NCGIA under grant number SES 88-10917 is gratefully acknowledged. Robert Franzosa helped us gain a better understanding of the theory of combinatorial topology. Students in our courses, especially David Steiner (Steiner, Egenhofer, and Frank, 1989), worked on implementations of the ideas reported here.

REFERENCES

Alexandroff, P., 1961, Elementary concepts of topology: Dover Publ., Inc., New York, 69 p.

Bertin, J., 1977, La graphique et le traitement graphique de l'information: Flammarion, Paris, 273 p.

Bertin, J., 1983, Semiology of graphics: Univ. Wisconsin Press, Madison, Wisconsin, 415 p.

Brodie, M., Mylopoulos, J., and Schmidt, J., eds., 1984, On conceptual modelling, perspectives from artificial intelligence, databases, and programming languages: Springer-Verlag, New York, 510 p.

Burrough, P., 1986, Principles of geographical information systems for land resources assessment: Oxford Univ. Press, Oxford, 193 p.

Buttenfield, B., ed., 1989, Multiple representations: initiative 3 specialist meeting report: Technical Report 89-3, National Center for Geographic Information and Analysis, Univ. California at Santa Barbara, 87 p.

Cardelli, L., 1984, A semantics of multiple inheritance, *in* Kahn, G., McQueen, D., and Plotkin, G., eds., Semantics of data types, Lecture Notes in Computer Science: Springer-Verlag, New York, v. 173, p. 51–67.

Carlson, E., 1987, Three dimensional conceptual modeling of subsurface structures: Proc. ASPRS-ACSM Annual Convention, v. 4, Baltimore, Maryland, p. 188–200.

Chan, K., and White, D., 1987, Map algebra: an object-oriented implementation: Proc. Intern. Geographical Information Systems (IGIS) Symposium: The Research Agenda, Arlington, Virginia, p. 127–150.

Chen, P. S. S., 1976, The entity-relationship model: towards a unified view of data: ACM Transactions on Database Systems, v. 1, no. 1, p. 9–36.

Clarke, K., 1990, Analytical and computer cartography: Prentice-Hall, Englewood Cliffs, New Jersey, 290 p.

Cleveland, W., and McGill, R., 1984, Graphical perception: theory, experimentation, and application to the development of graphical methods: Jour. Am. Stat. Assoc. v. 79, no. 387, p. 531–554.

Codd, E. F., 1970, A relational model for large shared data banks: Comm. ACM, v. 13, no. 6, p. 377–387.

Codd, E. F., 1972, Further normalization of the data base relational model, *in* Rustin, R., ed., Data base systems: Prentice-Hall, Englewood Cliffs, New Jersey, p. 33–64.

Corbett, J., 1979, Topological principles of cartography: Technical Report 48, Bur. Census, Dept. Commerce, Washington, D.C., 50 p.

Dahl, O.-J., and Nygaard, K., 1966, SIMULA—an algol-based simulation language: Comm. ACM, v. 9, no. 9, p. 671–678.

Date, C. J., 1986, An introduction to database systems: Addison-Wesley Publ. Co., Reading, Massachusetts, 574 p.

Dittrich, K., 1986, Object-oriented database systems: the notation and the issues, *in* Dittrich, K., and Dayal, U., eds., International Workshop in Object-Oriented Database Systems, Pacific Grove, California, p. 2–4.

Dorenbeck, C., and Egenhofer, M., 1991, Algebraic optimization of combined overlay operations: Proc. Tenth Intern. Symposium on Computer-Assisted Cartography, Baltimore, Maryland, p. 296–312.

Douglas, D., and Peucker, T., 1973, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature: The American Cartographer, v. 10, no. 2, p. 112–122.

Egenhofer, M., 1987, Appropriate concentual database schema designs for two-dimensional spatial structures: Proc. ASPRS-ACSM Annual Convention, v. 5, Baltimore, Maryland, p. 167–169.

Egenhofer, M., 1988, Graphical representation of spatial objects: an object-oriented view: Technical Report 83, Department of Surveying Engineering, University of Maine, Orono, Maine, 17 p.

Egenhofer, M., 1989, A formal definition of binary topology relationships, *in* Litwin, W., and Schek, H.-J., eds. Third Intern. Conference on Foundations of Data Organization and Algorithms, Lecture Notes in Computer Science: Springer-Verlag, New York, v. 367, p. 457–472.

Egenhofer, M., 1990, Interaction with geographic information systems via spatial queries: Jour. Visual Languages and Computing, v. 1, no. 4, p. 389–413.

Egenhofer, M., 1991a, Extending SQL for cartographic display: Cartography and Geographic Information Systems, v. 18, no. 4, p. 230–245.

Egenhofer, M., 1991b, Deficiencies of SQL as a GIS query language, *in* Mark, D., and Frank, A., eds., Cognitive and linguistic aspects of geographic space: Kluwer Academic Publ., Dordrecht, p. 477–491.

Egenhofer, M., and Frank, A. U., 1987, Object-oriented databases: database requirements for GIS: Proc. Intern. Geographic Information Systems (IGIS) Symposium: The Research Agenda, Arlington, Virginia, p. 189–211.

Egenhofer, M., and Frank, A. U., 1989, Object-oriented modeling in gis: inheritance and propagation: Proc. Ninth Intern. Symposium on Computer-Assisted Cartography, Baltimore, Maryland, p. 588–598.

Egenhofer, M., Frank, A. U., and Jackson, J., 1989, A topological data model for spatial databases, *in* Buchmann, A., Günther, O., Smith, T., and Wang, Y., eds., Symposium on the Design and Implementation of Large Spatial Databases, Lecture Notes in Computer Science: Springer-Verlag, New York, v. 409, p. 271–286.

Egenhofer, M., and Herring, J., 1991, High-level spatial data structure, *in* Maguire, D., Goodchild, M., and Rhind, D., eds., Geographical information systems: principles and applications: Longman, London, p. 227–237.

Ehlers, M., Edwards, G., and Bedard, Y., 1989, Integration of remote sensing with geographic information systems: a necessary evolution: Photogrammetric Engineering & Remote Sensing, v. 55, no. 11, p. 1619–1627.

Foley, J., van Dam, A., Feiner, S., and Hughes, J., 1990, Computer graphics: principles and practice: Addison–Wesley Publ. Co., Reading, Massachusetts, 430 p.

Frank, A. U., 1992, Spatial concepts, geometric data models and data structures: Computers & Geosciences, v. 18, no. 4, p. 409–417.

Frank, A. U., and Kuhn, W., 1986, Cell graph: a provable correct method for the storage of geometry: Proc. Second Intern. Symposium on Spatial Data Handling, Seattle, Washington, p. 411–436.

Freeman, H., and Ahn, J., 1987, On the problem of placing names in a geographic map: Intern. Jour. Pattern Recognition and Artificial Intelligence, v. 1, no. 1, p. 121–140.

Goldberg, A., and Robson, D., 1983, Smalltalk-80: Addison–Wesley Publ. Co., Reading, Massachusetts, 516 p.

Güting, R., 1988, Geo-relational algebra: a model and query language for geometric database systems, *in* Schmidt, J., Ceri, S., and Missikoff, M., eds., Advances in Database Technology—EDBT '88, Intern. Conference on Extending Database Technology, Venice, Italy, Lecture Notes in Computer Science: Springer-Verlag, New York, v. 303, p. 506–527.

Guttag, J., Horning, J., and Wing, J., 1985, The Larch family of specification languages: IEEE Software, v. 2, no. 4, p. 24–36.

Herring, J., 1987, TIGRIS: topologically integrated geographic information systems: Proc. Eighth Intern. Symposium of Computer-Assisted Cartography, Baltimore, Maryland, p. 282–291.

Herring, J., Larsen, R., and Shivakumar, J., 1988, Extensions to the SQL language to support spatial analysis in a topological data base: Proc. GIS/LIS '88, San Antonio, Texas, p. 741–750.

Imhof, E., 1972, Thematic cartography (in German): de Gruyter, Berlin, 270 p.

Ingram, K., and Phillips, W., 1987, Geographic information processing using a SQL-based query language, Proc. Eighth Intern. Symposium on Computer-Assisted Cartography, Baltimore, Maryland, p. 326–335.

Jackson, J., 1989, Algorithms for triangular irregular networks based on simplicial complex theory: Proc. ASPRS-ACSM Annual Convention, v. 4, Baltimore, Maryland, p. 131–136.

Khoshafian, S., and Abnous, R., 1990, Object orientation: concepts, languages, databases, user interfaces: John Wiley & Sons, New York, 434 p.

Kuhn, W., and Frank, A. U., 1991, A formalization of metaphors and image-schemas in user interfaces, *in* Mark, D., and Frank, A., eds., Cognitive and linguistic aspects of geographic space: Kluwer Academic Publ., Dordrecht, p. 419–434.

Lakoff, G., 1987, Women, fire, and dangerous things: what categories reveal about the mind: Univ. Chicago Press, Chicago, Illinois, 614 p.

Lakoff, G., and Johnson, M., 1980, Metaphors we live by: Univ. Chicago Press, Chicago, 242 p.

Mackinlay, J., 1986, Automating the design of graphical presentations of relational information: ACM Transactions on Graphics, v. 5, no. 2, p. 110–141.

Mark, D., Frank, A., Egenhofer, M., Freundschuh, S., McGranaghan, M., and White, R. M., eds., 1989, Languages of spatial relations: report on the specialist meeting for NCGIA research initiative 2: Technical Rept. 89-2, National Center for Geographical Information and Analysis, Univ. California at Santa Barbara, 62 p.

Meyer, B., 1988, Object-oriented software construction: Prentice-Hall, New York, 535 p.

Mounsey, H., and Tomlinson, R., eds., 1988, Building databases for global science: Taylor & Francis, London, 430 p.

Munkres, J., 1966, Elementary differential topology: Princeton Univ. Press, Princeton, New Jersey, 112 p.

National Research Council (NRC), 1980, Need for a multipurpose cadastre: National Academy Press, Washington, D.C., 112 p.

Robertson, P., 1988, Choosing data representations for the effective visualization of spatial data: Proc. Third Intern. Symposium on Spatial Data Handling, Sydney, Australia, p. 243–252.

Robinson, A., Sale, R., Morrison, J., and Muercke, P., 1984, Elements of cartography: John Wiley & Sons, New York, 448 p.

Roussopoulos, N., Faloutsos, C., and Sellis, T., 1988, An efficient pictorial database system for PSQL: IEEE Transactions on Software Engineering, v. 14, no. 5, p. 630–638.

Samet, H., 1989a, The design and analysis of spatial data structures: Addison–Wesley Publ. Co., Reading, Massachusetts, 493 p.

Samet, H., 1989b, Applications of spatial data structures: Computer Graphics, Image Processing, and GIS: Addison–Wesley Publ. Co., Reading, Massachusetts, 507 p.

Schlichtmann, H., 1984, Characteritic traits of the semiotic system "map symbolism": The Cartographic Jour., v. 22, p. 23–30.

Spanier, E., 1966, Algebraic topology: McGraw-Hill Book Co., New York, 528 p.

Steiner, D., Egenhofer, M., and Frank, A. U., 1989, An object-oriented carto-graphic output package: Proc. ASPRS-ACSM Annual Convention, v. 5, Baltimore, Maryland, p. 104–113.

Steinitz, C., Parker, P., and Jorden, L., 1976, Hand-drawn overlays: their history and prospective uses: Landscape Architecture, v. 66, no. 8, p. 444–455.

Stevens, S., 1946, On the theory of scales of measurement: Science Magazine, v. 103, no. 2684, p. 677–680.

Stroustrup, B., 1986, The C++ programming language: Addison-Wesley Publ. Co., Reading, Massachusetts, 328 p.

Tobler, W., 1979, A transformational view of cartography: The American Cartographer, v. 6, no. 2, p. 101–106.

Tomlin, C. D., 1990, Geographic information systems and cartographic modeling: Prentice-Hall, Englewood Cliffs, New Jersey, 249 p.

Tufte, E., 1983, The visual display of quantitative information: Graphics Press, Chesire, Connecticut, 197 p.

Tufte, E., 1990, Envisioning information: Graphics Press, Chesire, Connecticut, 126 p.

Ullman, J., 1982, Principles of database systems: Computer Science Press, Rockville, Maryland, 484 p.

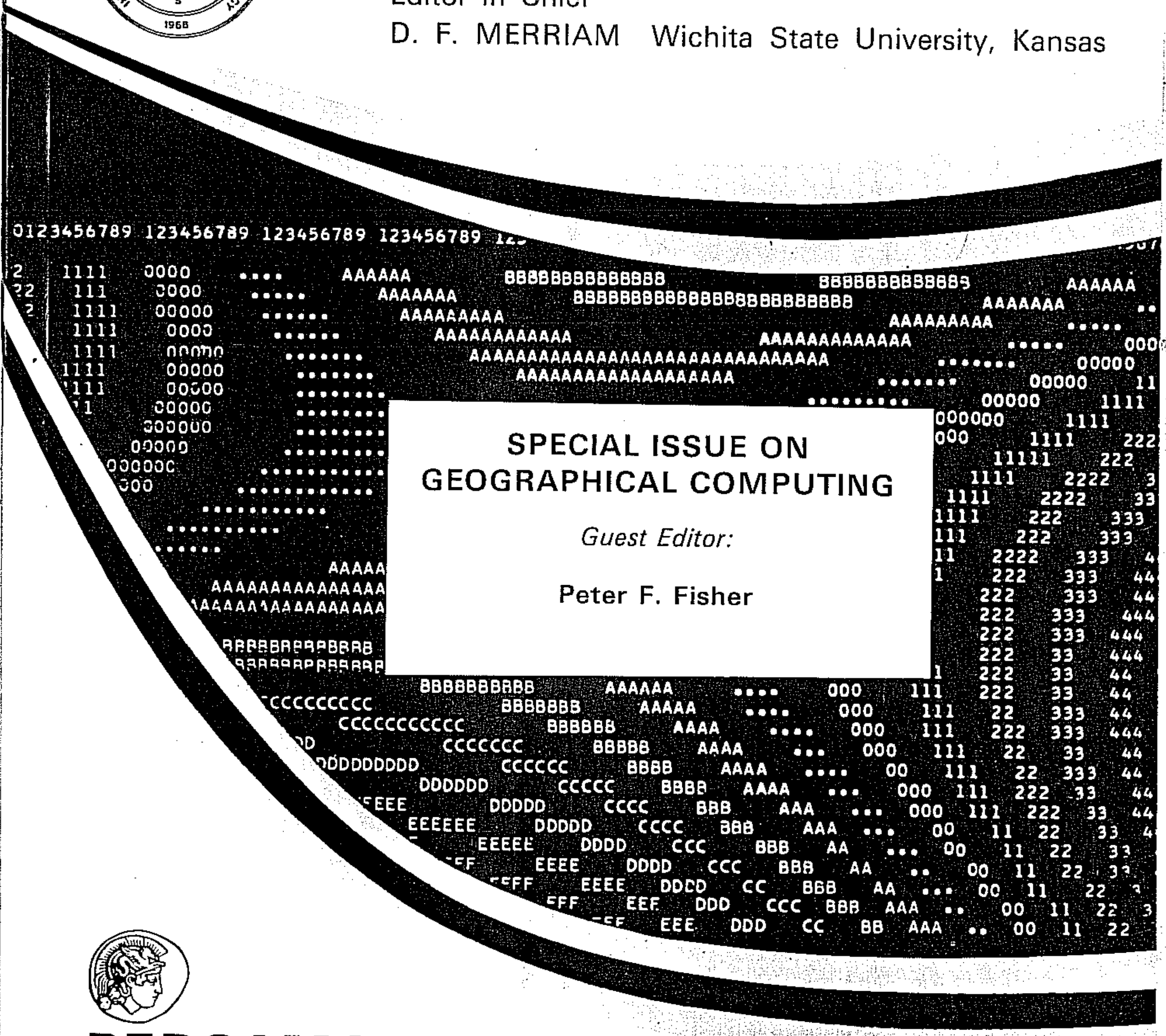
Worboys, M., Hearnshaw, H., and Maguire, D., 1990, Object-oriented data modelling for spatial databases: Intern. Jour. Geographical Information Systems, v. 4, no. 4, p. 369–383.

Zdonik S., and Maier, D., Fundamentals of object-oriented databases, *in* Zdonik, S., and Maier, D., eds., Readings in object-oriented database systems: Morgan Kaufmann Publ., Inc., San Mateo, California, p. 1–32.

Volume 18 Number 8 September 1992 ISSN 0098-3004

An International Journal

Editor-in-Chief
D. F. MERRIAM Wichita State University, Kansas



SPECIAL ISSUE ON GEOGRAPHICAL COMPUTING

Guest Editor:

Peter F. Fisher

PERGAMON PRESS
OXFORD · NEW YORK · SEOUL · TOKYO