A Full Function GIS Editor

William H. Moreland

Environmental Systems Research Institute
380 New York Street
Redlands, California 92373

ABSTRACT

With the introduction of engineering workstations, GIS graphic editors now have the means to the provide the user a single interface combining the capabilities of high speed graphics along with a fully functional two-way link to a relational database. A GIS editor must perform more than graphic edits, it must also maintain the integrity of a GIS database while editing both coordinates and attributes. The GIS editor needs to consider both the coordinates and attributes of each feature as a single entity; and to operate upon each entity with fast graphics operations as well as allowing full attribute editing capability. This paper will outline the requirements and specification of such an editor.

Introduction

The most common type of editors edit a single data type or file; but since a GIS database is not a single data type, but in fact a collection different data types which together form a database of both spatial and non-spatial data (figure 1). A person wishing to edit a GIS database does not want to edit any given part of the database as a single data type, but to alter the database as a whole. Therefore, a GIS editor must consider the GIS database as a whole and allow the user to alter it similarly. Most editors to date have performed well on either of the two main parts of a GIS database; the coordinates (spatial) or the attributes (non-spatial), but not both.

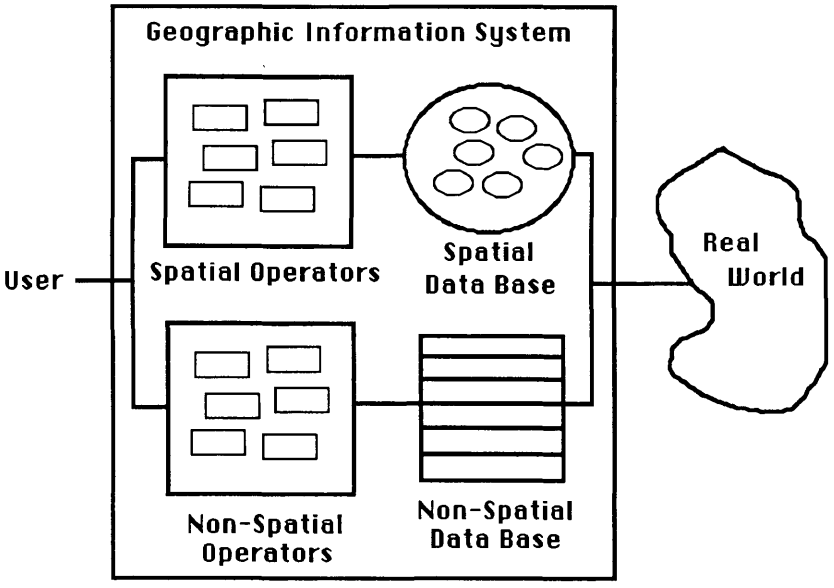


Figure 1.

There is a third component of a GIS that most editors seem to conveniently forget: topology. Topology is the glue that holds the GIS database together. The end result of most GIS editors is the edited versions of the two main components, that still need to be glued together. What is needed is an editor that allows fast efficient update of both of the spatial and non-spatial components while maintaining the topology. A fully functional GIS editor must produce a fully functional GIS database. What this paper will describe is how ARCEDIT has evolved over the years in response to the growing requirements for GIS editor, and our future plans for it.

A Full Function GIS Editor

A GIS editor needs to allow the user to alter and view any aspect of the low level components of the database. For ARC/INFO coverages these components are: Arcs, Nodes, Points, Annotation, and Tics (figure 2). The attributes (if any) of each of these components are linked with their counterparts to form a single entity capable of being edited (figure 3). This is ARCEDIT's basic purpose, since the coordinates and the attributes of each component are treated as a whole the editor not the user worries about the link between them and how that changes when either is altered. All the user cares about is that he has either moved the location or changed the value of an attribute.

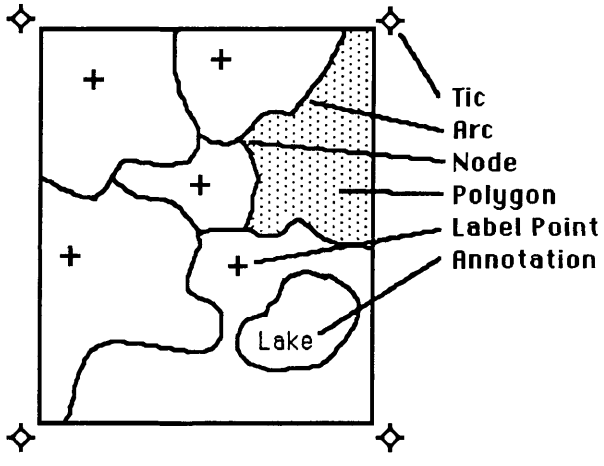


Figure 2 **(coverage)**.

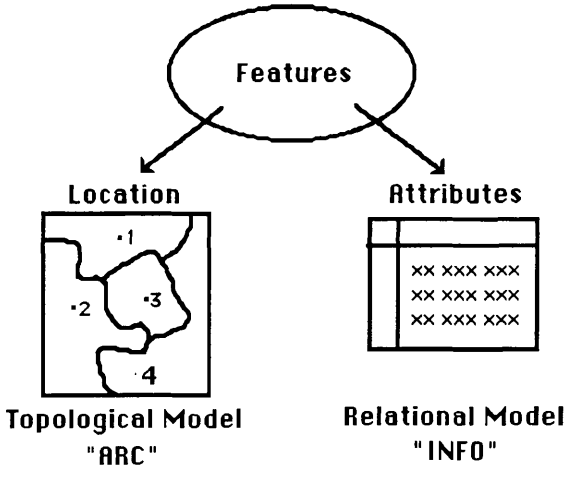


Figure 3.

A Full Function GIS Editor

As would be expected in a graphic editor, ARCEDIT allows any feature to be moved, copied,rotated, deleted, added, etc; as well as allowing any of the attributes to be calculated, assigned, etc. ARCEDIT satisfies the requirements of a GIS editor, by providing the user with environments to perform as much of the functions of the CLEAN and BUILD process as possible while maintaining an good response time.

CLEAN and BUILD are the topology builders within ARC/INFO. CLEAN takes as input the arcs and/or points (labels) finds all intersections between segments, and processes them into polygon or line coverages. BUILD and CLEAN perform basically the same operation, but BUILD skips the intersection phase in order to save time.

ARCEDIT has a number of editing environments that control how coordinates are handled while editing. These five environments: nodesnap, arcsnap, snapping, intersectarcs, and attribute only. These environments are responsible for ensuring that the nodes (end points of arcs) snap to their neighbors if within tolerance, resolving intersections within the arcs, snapping any component with any other from any other database, resolving both undershoots (arcs that are suppose to intersect but fall short) and overshoots (arcs that are suppose to meet exactly with another), and allowing non-spatial edits to not destroy the topology.

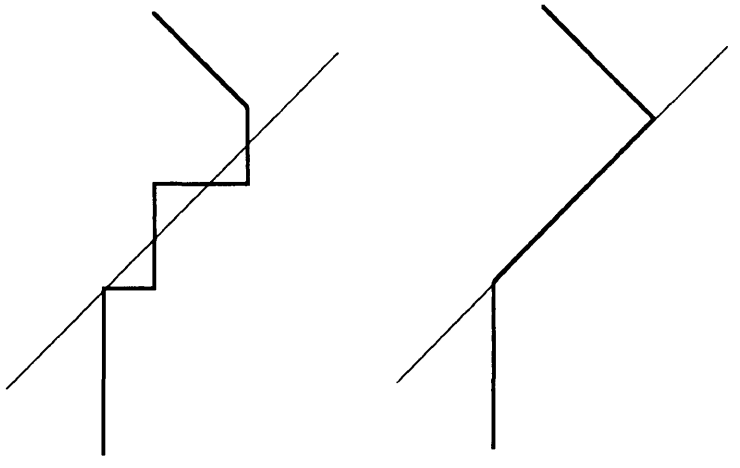
The snapping environments work in three ways. First all nodes are always snapped to each other when ever an arc is updated. This is to ensure that all arcs that are suppose to meet at a single node do in fact share the same coordinates for the common node. The second environment allows any feature being altered to be snapped to any other feature either within the same or any other coverage. This allow different feature types that are suppose to overlap and meet, to in fact do so and to have the same coordinates. The final snapping environment is where under and over shoots are resolved upon the adding of new arcs. This is used to ensure that those arcs that are suppose to end exactly at another do in fact do so and that where it does meet the other arc that, that arc is split and a node is generated. This is the only environment that is active only upon the adding of new arcs, and not during the entire edit session.

The intersection environment performs much of the preprocessing that goes on within CLEAN by ensuring that all the possible intersections between arcs have already been resolved and therefore it is not necessary to use CLEAN, but only reestablish topology by only using BUILD. Each arc as it is edited is checked against the region of the database that it overlaps for intersection with any existing arc, then the arc is checked against itself. Each new segment of the edited arc will each possess its own copy of the initial attributes.

The last environment is not really an environment at all, but in reality a internal flag that keeps tract of any spatial edits. If at the end of the edit session no spatial edits were performed, then only the non-spatial side of the database is altered and the topology of the spatial side remains intact. This allows Users to use the spatial capabilities of a GIS editor to select the components for update and to edit the attributes directly. This is where, once the database is built, most of the edits will take place. The user views the database as a whole and should be allowed to edit it as one; and not be forced to use one editor for spatial updates and another for non-spatial.

All of these environments are checked only when coordinates of any feature are updated and are order dependent. In the case of arcs, after each arc is edited, but before it is added back into the coverage, it is acted upon in the following order:

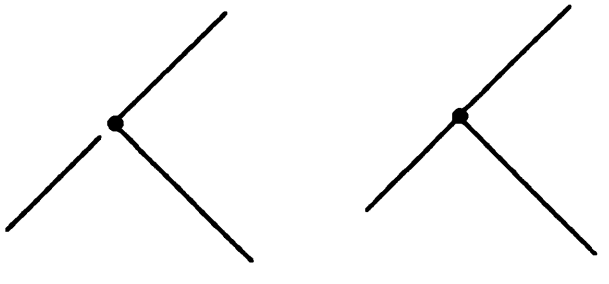
1) **SNAPPING** Check arc for snapping to any other feature of this or another coverage. This snapping environment is valid for all features, where as the rest are only valid for arcs.



SNAPPING

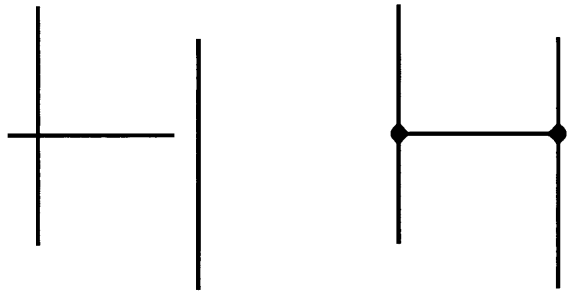
2) Check the nodes of the arc for snapping onto each other. In this case the last (to) is snapped onto the first (from). This ensures that islands (polygons represented by a single arc) are closed. This is not an environment, but does use the snap tolerance set by NODESNAP.

3) **NODESNAP** Check the nodes of the arc for snapping onto all of the existing nodes within the region of the coverage defined by the confining box of the altered arc. This environment is for arcs only.



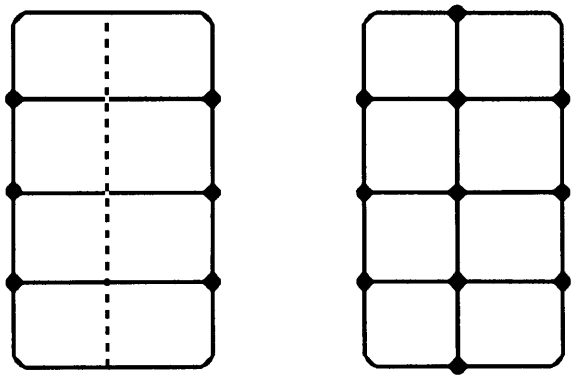
NODESNAP

4) **ARCSNAP** Check for under or over shoots. This environment is for arcs only.



ARCSNAP

5) **INTERSECTARCS** Check for intersections with either existing arcs or itself. This environment is for arcs only.



INTERSECTARCS

6) Recalculate the length of the arc and add to the coverage. This is not an environment, but is performed for every arc.

A Full Function GIS Editor

The non-spatial side of the database, as discussed earlier, should be fully accessible and changeable from a GIS editor. The user needs to be able to view and change any attribute of any feature, and to be able to establish selection sets by either attribute equation or by spatial selection. In other words, the user should be able to spatially alter any set of features derived by a attribute equation; and to be able to alter the attributes of any set of features derived by a spatial select. Since ARCEDIT considers the GIS database as a single entity in principle as well as in practice, it allows the user to select, view, and update any aspect of the database.

As with all editors, there needs to be a way of recovering from making mistakes. All updates to a database must not be irreversible, until such time as users save their updates. The philosophy of ARCEDIT is to only access the database in a Read-Only mode placing all updates into local scratch files. It is this philosophy that allows ARCEDIT to be able at any time, OOPS any number of updates all the way back to the beginning of the edit session, and to support recovery of updates in the case of the computer system going down.

The last thing all GIS editors needs to do is to maintain the topology. The topology is a part of the GIS database and therefore, should be accessible and maintainable by the editor. The above described editing environments go a long way toward accomplishing many of the low level steps required for maintaining topology interactively, but the final step of rebuilding the arc segments into polygons is not taken. We plan to tkae this final step in the next revision of ARCEDIT (ARC/INFO version 6.0). It is the introduction of engineering workstations, that make it possible to deliver topology on the fly while maintaining good interactive response time.

Conclusion

Basically a full function GIS should be able to work with the GIS database in its entirety and to present at all times an intact and fully functional GIS database to the user. This implies that upon completing the edit session the end result, if given a fully functional GIS database at the onset, is a fully functional GIS database. With the introduction of engineering workstations, and the steps already incorperated within ARCEDIT, there is enough compute power to ensure good response time while maintaining topology on the fly.