Spatial Data – the Final Frontier

(To boldly go into 3, 4 or more dimensions).

Mar 2009

This month's topic: 3D Validation

It is considered important that data that is captured, used and stored is valid. This is taken as a given, but the question of validation is far from simple. For example, incoming electronic survey plans may have 3D parcels included. The question of how to validate this information is highly significant.

Why Validate?

So the first issue to be discussed is why we want to validate data in the first place. There are several possibilities:

Processing Requirements

Invalid data can create difficulties for the software that is to use the data. In an extreme case, the database may not be able to accept the data at all. For example:

A non-numeric character in a field which is supposed to be numeric.

A Polygon with fewer than 3 points.

This second example raises a potential problem. It may be that two different DBMS's may have different rules in this case, and some may allow polygons with fewer than 3 points ("degenerate" cases).

On the other hand, it is important that validation is not being used to hide deficiencies in the software. Validation is expensive, and the correction of "errors" detected in validation is far more expensive. A validation check should only be allowed if the error it is attempting to detect is really an error.

It can be difficult to determine what restrictions exist in the software that is being used to process the data, and therefore this kind of validation is very problematic. For example, the re-factoring of a clip routine has caused a particular type of "pinched" polygon to be incorrectly handled. This condition was not identified in the clip specifications, but now a change in the validation routines is being used to prevent its occurrence. Manual correction of the occurrences will be needed.

In situations such as the digital lodgement of survey data, it is difficult to predict what use the data will be put to, and what software will be used, so the validation rules must be decided in a vacuum, and some attempt made to document them rigorously.

Data "Correctness"

Invalid data is probably incorrect data. The example of the non-numeric character highlights this. The polygon with fewer than 3 points is also probably incorrect if it intended to represent an area feature such as a lake.

But – this type of validation is the most difficult to define.

Firstly, it is impossible to ensure correctness by validation.

Secondly, it is probably impossible to think of all possible validation tests than can be applied.

Taking the polygon example, a polygon with three points may be valid, but this does not ensure the points are in the right place.

Furthermore invalid data is not necessarily incorrect. Any spatial data stored in a computer is an approximation to the real world situation, and "correct" in this context has to take account of this fact.

If the points in a polygon are close enough to be a good approximation, then the polygon is a correct representation. Polygons with "technical" problems, such as very small knots, miscloses, or mismatches with adjoining polygons can be correct (a good enough approximation), but invalid. What is more, these invalid polygons can be useful and valuable data for certain applications (such as providing a web map service).
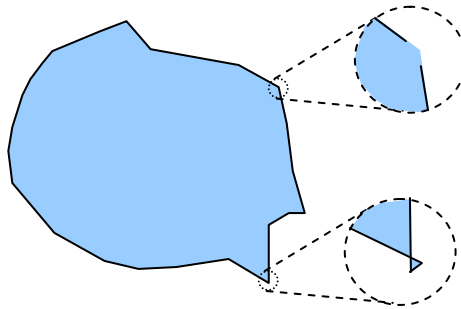

Figure 1 This polygon is an accurate representation of a lake, but is invalid because of a small knot, and a misclose

For some purposes, an object such as the lake pictured in Figure 1 is adequately represented, even though, for other purposes it is invalid. In any case, as a representation, it is correct.
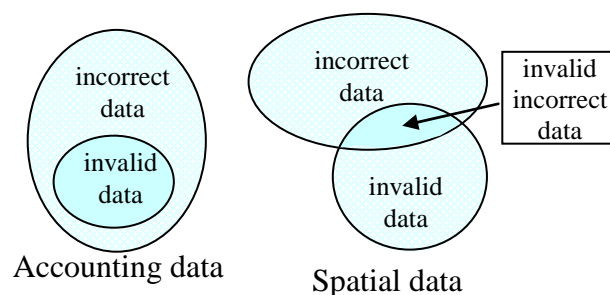

Figure 2 (Venn diagram) In more conventional forms of data, invalid data is always incorrect. In spatial data, there are many cases where correct data fails a validation test

Sales Promotion

This is an unusual and regrettable use of validation. For many years, in the GIS industry, it has been common for software salesmen to promote that their particular product can "find the errors xxx data" (where the xxx is a competitor's software

product). They should, of course, say "some of the errors", but the inference is that the more errors found the better.

The main problem with this approach is that the rules for validation are usually only described, and no formal definition of validity is given. For example, there may be pictures of "hanging lines" in the documentation, but no details of the actual definition of a hanging line.
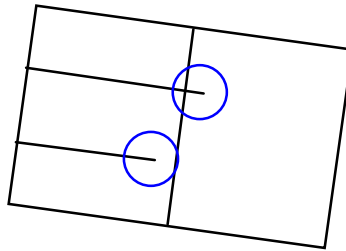

Figure 3 Hanging lines (circled) in cadastral data

2D Case

The issue of validating 2D spatial data has been being addressed for many years now, but is not yet solved. There are still questions to be resolved.

For example, the apparently simple case of a 2D polygon. It turns out that there are as many variations on what is a valid polygon as there are software manufacturers [1].

In an attempt to remedy this, the Open Geospatial Consortium (OGC) has developed a specification for simple features which gives a set of validation tests for simple polygons. This specification, which is widely accepted, makes a polygon with a hole touching the outer edge at two points, such as that in Figure 4 invalid. This is because it can be represented as two separated simple polygons – a multi-polygon.
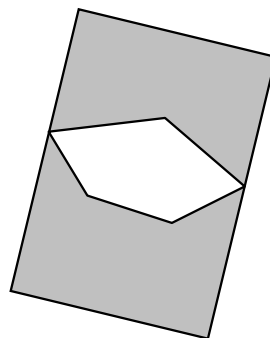

Figure 4 Invalid polygon with a hole

What the specification does not state is how to test for this case, and how to ensure that all software will agree on its invalidity.

Even with these specifications, transferring data between environments based on different software is a costly and time consuming business, which typically involves

3D Validation

the manual correction of "errors" determined by such validation tests. Some of the rules can seem inconsistent, since for example the rules for inner boundaries (holes), need not be the same as those for outer boundaries [2].

3D Case

So – what is needed in 3D?

Looking at the first criterion above, we need to be able to process the data that we validate. This means that we could base our validation on the method of storage that is to be used to support the data.

For example, if we were to use the Oracle DBMS to store 3D data, then incoming data is valid if Oracle says so. The sort of validation that Oracle spatial applies can be found in [3].

This is not an answer, because we need a more definite statement of what is and what is not valid, which does not depend on an external organisation (for example, if Oracle decides to change its validation rules), and which does not tie us into a particular software environment.

Let us look at some things we need to validate:

Complete Information

There is a certain amount of information needed to define a 3D object. For example, it is not possible to define an object by points and lines only.
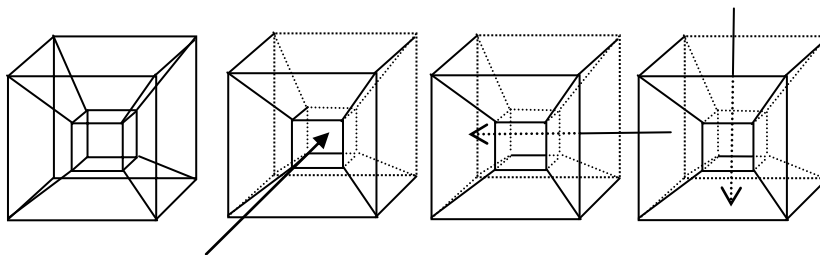

Figure 5 The object on the left is not uniquely defined. Without the faces, it can be interpreted in three different ways

It is necessary to completely close the object with faces in order to define the "inside" and the "outside". This is equivalent to saying in 2D, that a polygon must close, but is more difficult to determine.

In 2D, it is only necessary to ensure that the last point in a polygon coincides with the first point to be sure that a polygon closes, but there is no natural order in the faces of a 3D polyhedron, so that the computation is more difficult.

Planar Surfaces

If we have decided that faces will be used to define 3D objects, and they are intended to be planar, it is necessary to ensure that they really are planar.

For example, the volumetric cadastral parcel defining one of the overhangs of the Gabba cricket ground (Figure 6) is defined in terms of surfaces that are supposed to be planar.
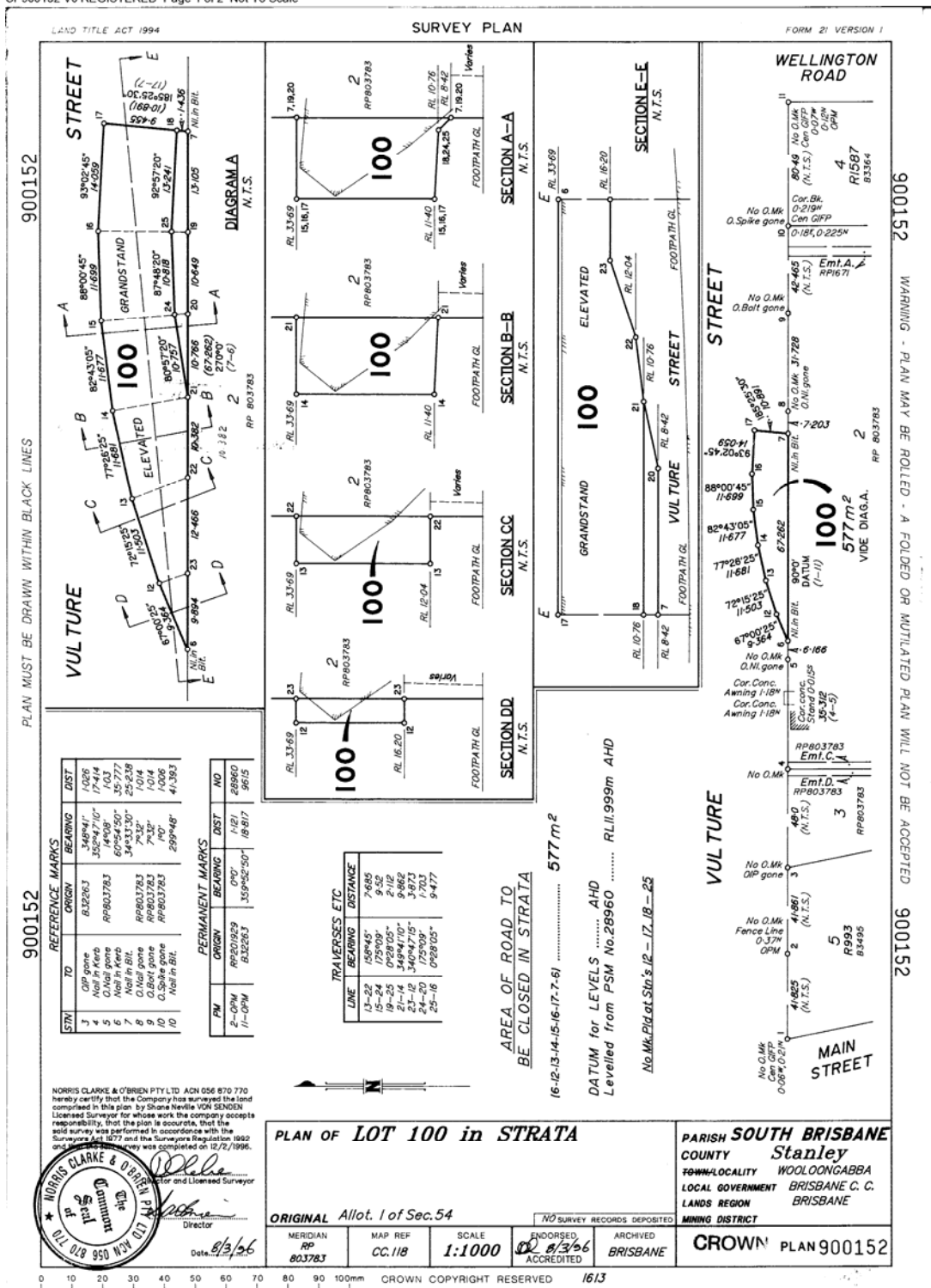
Figure 6 The Gabba stadium overhanging the road

Although the lower surfaces are not level, it is probably intended that they should be planar. They are not, according to the plan dimensions (e.g. consider the face 13,22,21,14 – cross sections C-C and B-B) . In this case, there is a real error in the data, but non-planarity of faces is often caused by rounding in the software.

If a surface is defined by three points, it can always be specified to be planar, but surfaces like the ones defining the bottom of the overhang in Figure 6 are defined by four or more points, and planarity cannot be guaranteed.

Does it Matter?

If a surface is supposed to be planar, but is not, there is a volume of space that may be inside or outside the boundary, depending on the interpretation taken.
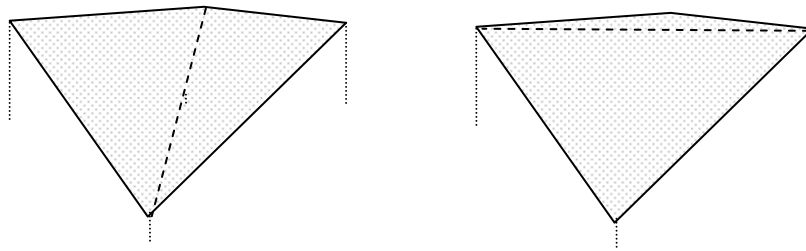

Figure 7 Attempting to define a surface by 4 points

In Figure 7, a surface is defined by four points, but it is not specified whether the shaded surface in the left figure is intended, or whether the alternate interpretation on the right, or whether some other surface is intended. Without this interpretation, it is impossible to calculate the volume enclosed by the surface.

Solutions

1. Allow a tolerance, but require that all surfaces are planar to within that tolerance. i.e. for tolerance t no point is more than t distance from the plane. (This is not simple, because even knowing that there is a plane within tolerance of the points does not define that plane. There is no guarantee that the receiving party will be able to calculate the plane).

2. Ensure that all surfaces are planar by ensuring they are all triangles. e.g. in Figure 7, make the dashed lines part of the definition.

3. Explicitly define the surface e.g. if it is a pane, then the plane can be defined by parameters A, B, C, D which define the surface to be the points for which Ax+By+Cz+D = 0.[4] (This approach is indicated if curved surfaces are allowed).

Watertight

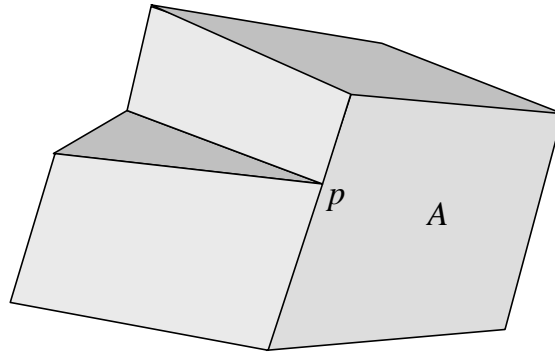It is important in certain software that the faces of a 3D object meet exactly.

Figure 8 It is important that there are no gaps between faces at *p*

For example, in Figure 8, A single face *A* meets three other faces at *p*. There is no need for *p* to be part of the definition of *A*, but if it is not, there may be a small sliver where the edges do not meet. On the other hand, if *p* is included as part of the definition of *A*, then the planarity of *A* needs to be ensured as discussed above.

Where non-planar surfaces are allowed – for example, in the Queensland cadastre, the issue is far more complex. This is an important issue in the the CAD/CAM (Computer Aided Drafting / Manufacture) world, but still the subject of research [5].

Non-Intersecting

In the same way that in 2D "knots" in polygon boundaries have to be avoided, it is important to ensure that the bounding surfaces of a 3D object do not self-intersect. The difference in 3D is that the nature of the self-intersection can be far more complex. The complicating factor both in 2D and 3D is that the self-intersection may be very small, and completely invisible at any reasonable scale.
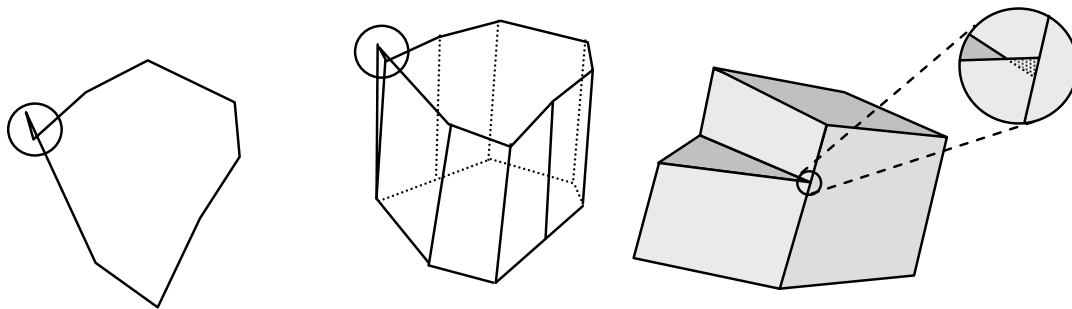

Figure 9 Knotted boundaries in 2D and 3D

In 3D, any attempt to remove these knots automatically will require far more complex algorithms. (And raises again the issue of non-planarity of faces).

Where non-planar surfaces are allowed – e.g. in the Queensland cadastre, the issue is again more complex. As an example, it is possible for curved surfaces to intersect in complex ways. In fact, there are quite simple surfaces which intersect in lines that cannot be defined parametrically.

Orientable

Finally, it is necessary to be aware that combinations of faces may not be orientable - that is to say, a surface should usually have two sides. In constructing a solid region in

space, it is usual that we need to define an inside and an outside, but in 3D, it is quite possible to define surfaces with only one side - The Möbius strip is the well-known example. This kind of surface cannot be used in constructing meaningful regions of space. (see http://mathworld.wolfram.com/MoebiusStrip.html).
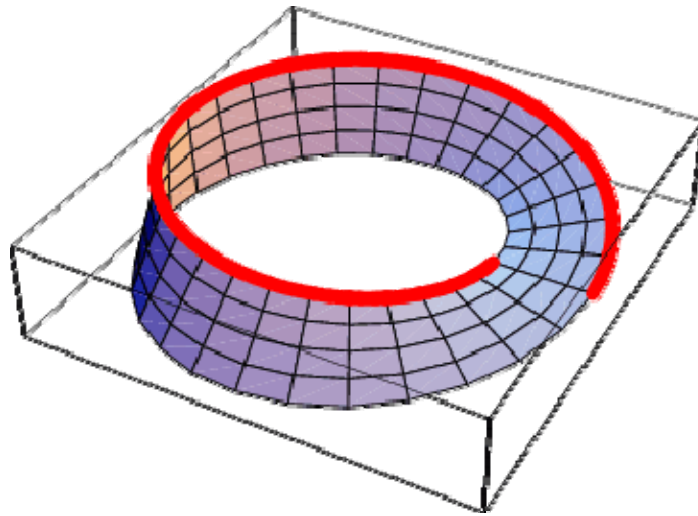

Figure 10 Möbius strip

Rod Thompson

Mar 2009

Conclusions

In deciding on validation to be applied to spatial data, it is important to consider why the validation is being done.

Validity does not ensure correctness.

Invalid data is not necessarily wrong!

It is important to determine what validation is really necessary, and not to be forced into a particular set of rules by software vendors, especially if those rules are not rigorously documented.

The writing of validation rules (even in 2D) is not simple.

The aim should be for a minimal set of validation rules. "The more errors detected the better" is not a good criterion.

Validation must not be used to hide software deficiencies.

For more details on specific issues in 2D and 3D object validation, see references [1] and [3] below. Both are available on the www.

References

1.   van Oosterom, P., W. Quak, and T. Tijssen (2004) *About Invalid, Valid and Clean Polygons*, in P.F. Fisher, (ed) *Developments In Spatial Data Handling*, Springer-Verlag: New York. p. 1-16.

http://www.gdmc.nl/publications/2004/Invalid_Valid_Clean_Polygons.pdf

2.	de Vries, M., W. Quak, and P. van Oosterom (2005) *Running a GML Relay; Interoperability Tests with Live Audience.* GIM International, **19**(12): p. 69-71.

3.	Kazar, B.M., R. Kothuri, P. van Oosterom, and S. Ravada (2008) *On Valid and Invalid Three-Dimensional Geometries*, in P. Van Oosterom, et al., (ed) *Advances in 3D Geoinformation Systems*, Springer: Berlin.

http://www.gdmc.nl/publications/2008/Valid_Invalid_3D_Geometries.pdf

4.	Thompson, R.J. (2007) *Towards a Rigorous Logic for Spatial Data Representation*, in *Geo Database Management Centre*. Delft University of Technology: Delft.

5.	Mukherjee, M., A.R. Daherkar, and R. Vattipalli (2000) *Satisfying coplanarity constraints on points in three dimensions with finite precision arithmetic.* Computer-Aided Design, **32**: p. 663-679.