OPENGIS PROJECT DOCUMENT 99-402r2

TITLE: Z values in Simple Features co-ordinates

AUTHOR: Name: Adam Gawne-Cain

Address: Cadcorp Ltd

Phone:

FAX:

Email: adam@cadcorpdev.co.uk

DATE: 15 August 1999

CATEGORY: Simple Features Revision Proposal

1. Background

There is a requirement for handling height information in geographic information systems, in a way that falls short of full 3D modeling. This simple method treats Z values (e.g. heights) as attributes of 2D vertices.

This document:
1) Defines the behavior of Z-Geometry in abstract terms
2) Suggests how Simple Features could be extended for Z values

This document defines how geometry can have Z values in a way which is a simple extension of the existing 2D OpenGIS Simple Features specification.

It does NOT cover full 3D geometry (e.g. solids), or 3D coordinate systems. Handling these topics thoroughly is a more complicated job, and is outside the scope of this document.

2. References

3. Proposal

3.1 Abstract

3.1.1 Definition of Z-Geometry

a) Z values are associated with every vertex.
b) Z values can be linearly interpolated between consecutive curve vertices.
c) Z values are undefined within surface interiors.

Any Z-Geometry can be converted into a 2D geometry by discarding all its Z values. The resulting 2D geometry is the “shadow” of the Z-Geometry. 2D geometries cannot be safely converted into Z-Geometries, since their Z values are undefined, and not necessarily zero.

3.1.2 Semantics of Z value

The Z value can be used to model any continuously varying field value that can be approximated by linear interpolation between 2D vertex positions.

The Z value at each vertex will often represent the height of the vertex. If this is the case, then the units of the height measurement can be specified as meters, feet, or any other linear unit.

The correct place to specify the units of the Z values is probably in the spatial reference system of the geometry. However, extending the spatial reference system to 3D is being addressed by the Coordinate Transformation RFP, and is outside the scope of this document.

While we wait for the CT RFP, we suggest the following Z units for applications that wish to use the Z values for height:
a) If the 2D SRS is a geographic coordinate system, then Z values are in meters.
b) If the 2D SRS is a projected coordinate system, then the Z values are homogenous with X and Y.

3.1.3 Simple Geometry

The OpenGIS Simple Features specification defines the term “simple geometry”. In general, a Z-Geometry is simple if it contains no self-intersection.

The term “simple geometry” is important, since Spatial Relationships and Spatial Operators are undefined on non-simple geometries.

A Z-Geometry is simple if

1) Its 2D shadow is simple
2) There is at most one Z value at each 2D location: (So wherever two of the geometry’s vertices match in (X,Y), they also match in Z. For example, this rule applies to the end-points of a LinearRing, and the tangential points of a multi-looped Polygon. Also, wherever the (X,Y) shadow of one of the geometry’s vertices lies in the interior of the (X,Y) shadow of another of the geometry’s line segments, then the vertex Z value matches the line segments interpolated Z value. This rule applies to the tangential point of a multi-looped polygon.)

With this definition, a sloping road centerline is simple, but a vertical flagpole is not. The fact that a flagpole is not simple is one of the limitations of Z-Geometries, whereas in a 3D system the flagpole could be considered simple.

An implementation of Z-Geometries could represent a flagpole, or a line-string with a vertical jump. But these Z-Geometries would not be simple, so the theoretically correct result of spatial operations would be undefined.

3.1.4 Spatial Relationships

The OpenGIS Simple Features specification defines spatial relationships in terms of a dimensionally extended nine-intersection matrix (DE9-IM).

The DE9IM rules for spatial queries are applied to the 2D shadows of all Z-Geometries. So a 3D point is inside a Z-Polygon if its (X,Y) location is inside the polygon’s 2D shadow. (For example, this allows testing whether a 3D GPS location is inside a land-parcel that is defined as a loop of 3D points.)

Spatial Operators

The OpenGIS Simple Features specification defines various spatial operators, which can be used to generate new geometries from existing geometries.

The operators Union, Intersect, SymmetricDifference and Difference are defined on Z-Geometries by working on their 2D shadows. This will produce a 2D output. However, the 2D output can be converted back to 3D by assigning a Z value to each output vertex. This can be done in a well-defined way, since each output vertex will either match an input vertex, or it will lie on a curve of an input geometry, where the Z is defined using linear interpolation. If an output vertex lies on both of the input geometries, then the "this" geometry object is used for the Z value.

The output of operators on mixed 2D and Z-Geometries will be 2D.

The output of the convex-hull operator on a Z-Geometry takes its Z values from the original Z-Geometry. Each vertex in the convex-hull output will correspond to a vertex in the input geometry, so its Z-value is well defined for simple geometries.

The buffer operator will work on the 2D shadow of Z-Geometries. Each vertex in the output geometry will be given a Z value equal to the smallest Z value in the input Z-Geometry.

Compatibility with existing OpenGIS Simple Features

3.1.5 Compatibility of Abstract Model

The geometries defined in the existing OpenGIS Simple Features specification are all two-dimensional (2D).

The abstract section of this document carefully defines spatial relationships and spatial operators on Z-Geometries, to ensure backwards-compatibility with Simple Features.

Backwards-compatibility at the abstract level can be shown with a mathematical argument along the following lines:

Define geometrical systems G, H and SF where

G = {Simple Z-Geometries }
H = {Simple Z-Geometries where all Z values are 0}
SF = {Simple 2D geometries, as defined by the OpenGIS Simple Features}

Then H is a subsystem of G, and H is isomorphic to SF, preserving the spatial relationship functions and the spatial operator functions.

3.1.6 Extensions to Well-Known-Binary format

The existing OpenGIS Simple Features specification has the following 2D geometry types:

```
enum wkbGeometryType {
      wkbPoint = 1,
      wkbLineString = 2,
      wkbPolygon = 3,
      wkbMultiPoint = 4,
      wkbMultiLineString = 5,
      wkbMultiPolygon = 6,
		wkbGeometryCollection = 7
	}
```

This type-code is put in the header of a WKB blob. To reflect that all of these 2D geometry types have a Z-analogue, we would add a flag to the type-code to indicate that all following points include a Z value. So the new enumeration would become:

```
	enum wkbGeometryTypeZ {
		wkbPoint = 1,
		wkbLineString = 2,
		wkbPolygon = 3,
		wkbMultiPoint = 4,
		wkbMultiLineString = 5,
		wkbMultiPolygon = 6,
		wkbGeometryCollection = 7,
		wkbZ = 0x80000000
	}
```

Alternatively, this could be less efficiently coded as:

```
	enum wkbGeometryTypeZ {
		wkbPoint = 1,
		wkbLineString = 2,
		wkbPolygon = 3,
		wkbMultiPoint = 4,
		wkbMultiLineString = 5,
		wkbMultiPolygon = 6,
		wkbGeometryCollection = 7,
		wkbPointZ = 0x80000001,
		wkbLineStringZ = 0x80000002,
		wkbPolygonZ = 0x80000003,
		wkbMultiPointZ = 0x80000004,
		wkbMultiLineStringZ = 0x80000005,
		wkbMultiPolygonZ = 0x80000006,
		wkbGeometryCollectionZ = 0x80000007,
	}
```

For example, a Z-Point geometry at the location (10,20,30) would be:

```
	WKBPoint {
		byte		byteOrder;	// wkbXDR or wkbNDR
		uint32		wkbType;	// (wkbPoint+wkbZ) =
	0x80000001
		Point {
			Double	x;		// 10.0
			Double	y;		// 20.0
			Double	z;		// 30.0
		}
	}
```

3.1.7 Extensions to Well-Known-Text format

A Z-Point at (10,20,30) would be represented as “POINT(10 20 30)”.

A Z-Linestring from (10,20,30) to (40,50,60) would be represented as “LINESTRING(10 20 30,40 50 60)”.

Notice that in the line-string, each vertex is terminated by either a comma, or a bracket. This means that geometries with Z values can be clearly distinguished from plain 2D geometries. However, if any vertex in a geometry has a Z value, then all the other vertices should have Z values. This keeps WKT in-step with WKB, where the presence or absence of Z values is marked by a single flag for the whole geometry.

3.1.8 Extensions to Microsoft COM Simple Features interfaces

Since we have not needed to introduce any new geometry classes, and we have not changed the abstract behavior of any existing 2D geometry objects, we should not need to change any existing COM Interface Definition Language (IDL) files.

All we need to do is add a new optional IDL file for Z values:

```
      // File: ZIdl.idl
      // Extension to Simple Features for Z values.
      import "ocidl.idl", "GeometryIdl.idl";

      interface IPointZ;

      [ object, uuid(D4579E2D-1D2B-11d3-80BE-00C04F680FFF) ]
      interface IPointZ : IPoint
      {
       [propget] HRESULT Z([out, retval] double * z);
      };
```

Since the new interface IPointZ inherits from IPoint, whenever a Z-aware component is asked for a point value (e.g. when getting a vertex from a line-string), the component can return an IPointZ interface pointer.

In Visual Basic, a Z-aware client which knows it is interoperating with a Z-aware server, could ask for the point locations with an IPointZ typed variable, and then pass this IPointZ variable back into other methods. For example:

```
    Dim line As ILineString
    Set line = CreateFromWKT("LINESTRING(10 11 12,20 21,30 31 32)",
sref)
    Dim sr As ISpatialRelation
    Set sr = line
    For i = 0 To 2
        ' Get line vertex as 3D point.
        Dim ptz As IPointZ
        Set ptz = line.Point(i)

        ' Re-use 3D point in other Simple Features methods.
        Debug.Print i; ") "; ptz.x, ptz.y, ptz.Z
        Debug.Print sr.Contains(ptz)
    Next i
```