Three-dimensional geometries in geoinformation

Lesson 3.2

To facilitate and encourage the exchange and interoperability of geographical information, the ISO (International Organization for Standardization: www.iso.org) and the OGC (Open Geospatial Consortium: www.opengeospatial.org) have developed in recent years standards that define what the basic geographical primitives are (the abstract specifications of ISO 19107), and also how they can be represented in a computer (the implementation specifications of GML and *Simple Features*). While the abstract definitions for the primitives are not restricted to two dimensions (2D), most of the efforts for the representation and storage of the geographical primitives have been done only in 2D; the *Simple Features* specifications are well-defined, used, and implemented across the GIS community.

Simple Features specifications

This document gives an overview of the primitives in 3D, both from the ISO 19107 and the GML point-of-views. Although the topic might appear trivial—"a polyhedron is simply a polyhedron, no?"—it is in practice a problem because several definitions exist and different software packages use different ones.

polyhedron

Having unambiguous definitions for the geometric primitives is important to foster interoperability, because most GIS operations (eg calculation of the area of polygons; creation of buffers; conversion to other formats; Boolean operations such as intersection, union, etc.) require that the input primitives be according to certain definitions, otherwise the output of the operation is not guaranteed.

1.1 Are your polyhedra the same as my polyhedra?

In the scientific literature, there is no single definition for a solid or a polyhedron (notice that these two terms are often used interchangeably). Even in the field of mathematics, opinions differ as to what constitutes the term *polyhedron*; many simply characterise the term as "difficult to define". Some researchers use it only for a regular polyhedron, or only for a convex one, and some consider non-planar faces as part of the definition.

The most common definition used is probably this simple one: a polyhedron is a 3D solid bounded by planar faces. The bounding faces are surfaces embedded in $\mathbb{R}^3$, the three-dimensional Euclidean space, and together the bounding surfaces form a *closed two-dimensional manifold* (or 2-manifold for short). A 2-manifold is a topological space that is topologically equivalent to $\mathbb{R}^2$. An obvious example is the surface of the Earth, for which near to every point the surrounding area is topologically equivalent to a plane. An example of a 3-manifold is the entire Earth (its interior) because the neighbourhood of every point is equivalent

2-manifold

(*last update*: *March* 8, 2021)

3-manifold

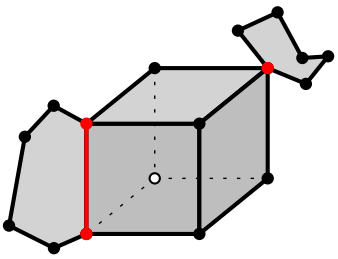

Figure 1.1: An example of an invalid 2-manifold: one edge and one vertex are non-manifold (the red ones).

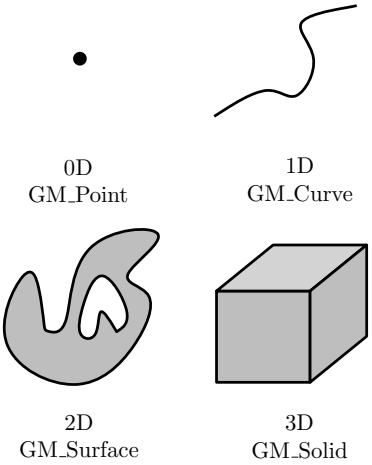

Figure 1.2: ISO 19017 primitives relevant for the modelling of the built environment.

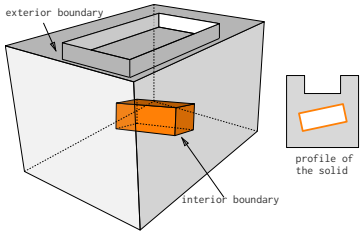

Figure 1.3: One solid which respects the ISO 19107 definition. It has one exterior shell (grey) and one interior shell (orange) forming a cavity.

to a sphere. The concept of neighbourhood, or locality, is such that a manifold can actually be constructed by 'gluing' separate Euclidean spaces together. Representing and storing a 2-manifold, even in $\mathbb{R}^3$, can be done with data structures that are intrinsically 2D since: (1) each edge is guaranteed to have a maximum of two incident faces; (2) around each vertex the incident faces form one 'umbrella' (Figure 1.1). The 2D data structures typically used in GIS, eg the half-edge or the DCEL, can thus be used.

1.2 The standard ISO 19107

The geometric primitives as used in 3D GIS are based on the ISO 19107 definitions, and the definition of a polyhedra there is broader than that of a 2-manifold, to allow us to represent all the real-world features.

As shown in Figure 1.2, the ISO 19107 geometric primitives for representing an object are: a 0D primitive is a `GM_Point`, a 1D a `GM_Curve`, a 2D a `GM_Surface`, and a 3D a `GM_Solid`. A d-dimensional primitive is built with a set of $(d-1)$-dimensional primitives, eg a `GM_Solid` is formed by several `GM_Surfaces`, which are formed of several `GM_Curves`, which are themselves formed of `GM_Point`. Observe that the ISO19107 primitives do not need to be linear or planar, ie curves defined by mathematical functions are allowed

In our context, the following three definitions from ISO (2003) are relevant:

Definition 1.2.1 *A `GM_Solid` is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces. The boundaries of `GM_Solids` shall be represented as `GM_SolidBoundary`. [...] The `GM_OrientablesSurfaces` that bound a solid shall be oriented outward.*

Definition 1.2.2 *A `GM_Shell` is used to represent a single connected component of a `GM_SolidBoundary`. It consists of a number of references to `GM_OrientableSurfaces` connected in a topological cycle (an object whose boundary is empty). [...] Like `GM_Rings`, `GM_Shells` are simple.*

Definition 1.2.3 *A `GM_Object` is* simple *if it has no interior point of self-intersection or self-tangency. In mathematical formalisms, this means that every point in the interior of the object must have a metric neighbourhood whose intersection with the object is isomorphic to an n-sphere, where n is the dimension of this `GM_Object`.*

Observe that since shells (`GM_Shells`) are *simple*, they are 2-manifold objects. To be a valid shell, the 2-manifold should be closed, ie there should not be 'holes' in the surface (in other words, it should be watertight).

Figure 1.3 shows a solid that respects that definition. First observe that the solid is composed of two shells (both forming its boundaries), one being the exterior and one being the interior shell. The exterior shell has eleven surfaces, and the interior one six. An interior shell creates a cavity in the solid—cavities are also referred to as "voids" or holes in a solid.

A solid can have no inner shells, or several. Observe that a cavity is not the same as a hole in a torus (a doughnut) such as that in Figure 1.4: it can be represented with one exterior shell having a genus of 1 and no interior shell. Observe also that the top face of the solid in Figure 1.3 has one inner ring, but that other surfaces "fill" that hole so that the exterior shell is closed.

The **genus** of an (orientable) surface embedded is the number of "handles" that it has. For instance, a doughnut and a mug have a genus of 1.

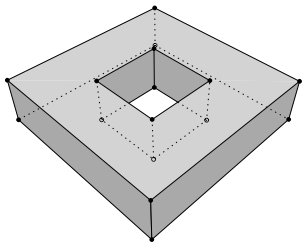

Figure 1.4: A 'squared torus' is modelled with one exterior boundary formed of ten surfaces. Notice that there are no interior boundary.

1.3 Primitives used in practice

CityGML, the international standard for 3D modelling of cities (see Lesson 4.1), uses a subset of ISO 19107, with the following two restrictions:

1. `GM_Curves` can only be *linear* (thus only `LineStrings` and `LinearRings` are used);
2. `GM_Surfaces` can only be *planar* (thus `Polygons` are used).

Following ISO 19107, in GML and CityGML geometric primitives can be combined into either *aggregates* or *composites*.

An aggregate (class `gml:_AbstractGeometricAggregate`) is an arbitrary collection of primitives of same dimensionality that is simply used to bundle together geometries. GML (and CityGML) has classes for each dimensionality (`Multi*`), the most relevant one in our context is `MultiSurface` that is often used in practice to represent the geometry of a building. An aggregate does not prescribe any topological relationships between the primitives.

A composite of dimension d is a collection of d-dimensional primitives that form a d-manifold. The most relevant example in our context is a `CompositeSurface`, which is a 2-manifold.

d-manifold

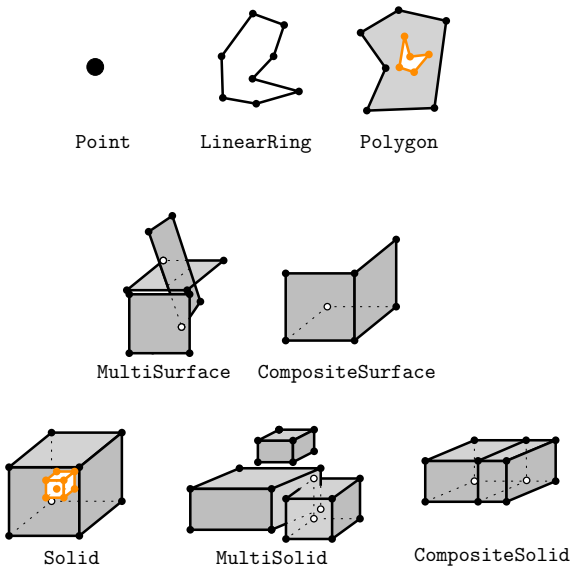

Figure 1.5: Some of the CityGML primitives, including aggregates and composites. Orange primitives are those representing inner boundaries. The `Shell` is not a class in GML, but it is implied when a `CompositeSurface` is used to define the boundary of a `Solid`.

1.4 Implementation specifications for the 3D primitives

Observe that for a primitive to be valid, all its lower-dimensionality primitives have to be valid. For instance, a valid Solid cannot have as one of its surfaces a Polygon having a self-intersection (which would make it invalid).

1.4.1 `Polygon`

For a `Polygon` embedded in $\mathbb{R}^3$ to be valid, it needs to fulfil the 6 assertions in Figure 1.6, which are given on pages 27–28 of the OGC *Simple Features* document. These rules are verified by first projecting each `Polygon` to a plane, this plane is usually obtained by least-square adjustment of its points. A `Polygon` must also be *planar* to be valid: its points (used for both the exterior and interior rings) have to lie on a plane.

1. Polygons are topologically closed;
2. The boundary of a Polygon consists of a set of LinearRings that make up its exterior and interior boundaries;
3. No two Rings in the boundary cross and the Rings in the boundary of a Polygon may intersect at a Point but only as a tangent, eg

$$\forall P \in Polygon, \forall c1, c2 \in P.Boundary(), c1 \neq c2,$$

$$\forall p, q \in Point, p, q \in c1, p \neq q, [p \in c2 \Rightarrow q \notin c2];$$

4. A Polygon may not have cut lines, spikes or punctures eg:

$$\forall P \in Polygon, P = P.Interior.Closure;$$

5. The interior of every Polygon is a connected point set;
6. The exterior of a Polygon with 1 or more holes is not connected. Each hole defines a connected component of the exterior.

Figure 1.6: The six assertions for the validity of a 2D polygon, according to *Simple Features*.

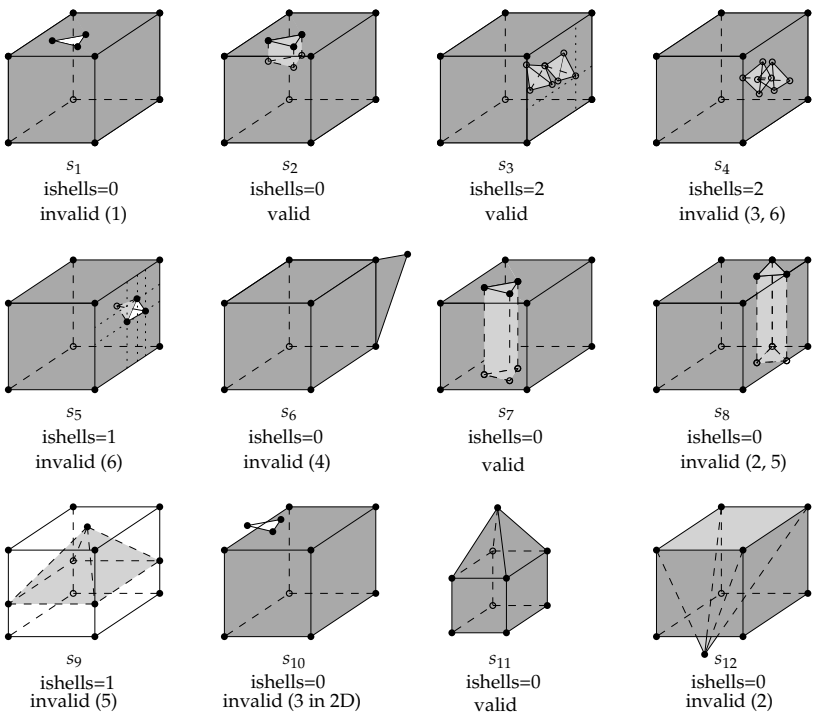

Figure 1.7: Twelve solids, some of them valid some invalid. The number of interior shell(s) is "ishell", and the numbers in parentheses next to invalid indicates which OGC assertions are broken. For solid s_9 the colour of the exterior shell is not shown to highlight the interior shell.

1.4.2 `MultiSurface`

It is an arbitrary collection of `Polygon`. Validating a `MultiSurface` simply means that each `Polygon` is validated individually; a `MultiSurface` is valid if all its `Polygons` are valid.

1.4.3 `CompositeSurface`

Besides that each `Polygon` must be individually valid, the `Polygons` forming a `CompositeSurface` are not allowed to overlap and/or to be disjoint. Furthermore, if we store a `CompositeSurface` in a data structure, each edge is guaranteed to have a maximum of two incident surfaces (except those on the boundary), and around each vertex the incident faces form one "umbrella" (see Figure 1.1).

1.4.4 `Solid`

According to ISO 19107, the different boundaries of a solid are allowed to interact with each other, but only under certain circumstances. To understand these, we have to generalise to 3D the implementation specifications defined in 2D by the OGC (Figure 1.6). Observe that all of them, except the third one, generalise directly to 3D since a point-set topology nomenclature is used. The only modifications needed are that, in 3D, polygons become solids, rings become shells, and holes become cavities.

To further explain what the assertions are in 3D, Figure 1.7 shows 12 solids, some of them valid, some not (all the statements below refer to solids in this figure).

The first assertion of the OGC means that a solid must be closed, or 'watertight' (even if it contains interior shells). The solid s_1 is thus not valid, but s_2 is because the hole in the top surface is 'filled' with other faces.

The second assertion implies that each shell must be simple, ie that it is a 2-manifold.

The third assertion means that the boundaries of shells can intersect each others, but the intersection between the shells can only contain primitives of dimensionality 0 (vertices) and 1 (edges). If a surface or a volume is part of the intersection, then the solid is invalid. The solid s_3 is an example of a valid solid: it has two interior shells whose boundaries intersect at one point (at the apexes of the tetrahedra), and the apex of one of the tetrahedra is coplanar with the exterior shell. If the interior of the two interior shells intersects (as in s_4) the solid is not valid; this is also related to the sixth assertion stating that each cavity must define one connected component: if the interior of two cavities are intersecting they define the same connected component. Notice also that s_5 is not valid since one surface of its cavity intersects with one surface of the exterior shell (they "share a surface"); s_5 should be represented with one single exterior shell (having a 'dent'), and no interior shell.

regularisation

The fourth assertion states that a shell is a 2-manifold and that no dangling pieces can exist (such as that of s_6); it is equivalent to the *regularisation* of a point-set in $\mathbb{R}^3$.

The fifth assertion states that the interior of a solid must form a connected point-set (in $\mathbb{R}^3$). Consider the solid s_7, it is valid since its interior is connected and it fulfils the other assertions; notice that: (1) it is a 2-manifold but that unlike other solids in Figure 1.7 (except s_8) its genus is 1; (2) it is modelled only with an exterior shell. If we move the location of the triangular prism (which is part of the exterior shell, and is not an interior shell) so that it touches the boundary of the exterior shell (as in s_8), then the solid becomes invalid since its interior is not connected anymore, and also since its exterior shell is not simple anymore (2 edges have 4 incident planar faces, which is not 2-manifold). It is also possible that the interior shell of a solid separates the solid into two parts: the interior shell of s_9 is a pyramid having four of its edges intersecting with the exterior shell, but no two surfaces are shared, thus these interactions are allowed. However, the presence of the pyramid separates the interior of the solid into two unconnected volumes (violating assertion 5); for both s_8 and s_9, the only possible valid representation is with two different solids.

Notice also that for a solid to be valid, all its lower-dimensionality primitives must be valid. That is, each surface of the shells has to be individually valid according to the assertions in Figure 1.6. An example of an invalid surface would be one having a hole (an inner ring) overlapping the exterior ring (see s_{10}).

combinatorial consistency

It should also be noticed that when validating a solid both the combinatorial consistency and the geometric consistency of the representation should be valid. A solid such as s_{11} is valid, but if the location of only one of its vertices is modified (for instance if the apex of the pyramid of s_{11} is moved downwards to form s_{12}) then it becomes invalid. Both

s_{11} and s_{12} can be represented with a graph having exactly the same topology (which is valid for both), but if we consider the geometry then the latter solid is not valid since its exterior shell is not simple. Enforcing simplicity requires calculating the intersections between the surfaces.

Lastly, the orientation of the polygons must be considered. In 2D, the only requirement for a polygon is that its exterior ring must have the opposite orientation of that of its interior ring(s) (eg clockwise versus counter-clockwise). In 3D, if one polygon is used to construct a shell, its exterior ring must be oriented in such as way that when viewed from outside the shell the points are ordered counter-clockwise. Figure 1.8 shows an example. In other words, the normal of the surface must point outwards if a right-hand system is used, ie when the ordering of points follows the direction of rotation of the curled fingers of the right hand, then the thumb points towards the outside. If the polygon has interior rings, then these have to be ordered clockwise.

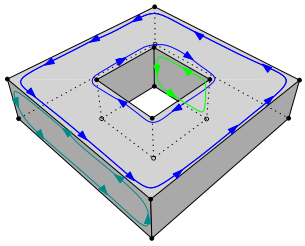

Figure 1.8: One solid and the orientation of 3 of its polygons (different colours).

⚙ How does it work in practice?

The software 'val3dity', developed at TU Delft, allows us to validate directly all the ISO 19107 primitives, it accepts as input CityJSON and OBJ, among others. It is freely available at `https://github.com/tudelft3d/val3dity`, and a web-application can be used at `http://geovalidation.bk.tudelft.nl/val3dity/`

1.4.5 `MultiSolid`

It is an arbitrary collection of `Solids`. Validating a `MultiSolid` simply means that each `Solid` is validated individually; a `MultiSolid` is valid if all its `Solids` are valid.

1.4.6 `CompositeSolid`

A `CompositeSolid`, formed by the `Solids` A and B, should fulfil the following two assertions:

- **Assertion #1:** their interior should not overlap ($A^o \cap B^o = \emptyset$)
- **Assertion #2:** their union should form one solid ($A \cup B$ = one `Solid`)

1.5 Exercises

1. List all 10 surfaces (and describe their geometry) for the solid in Figure 1.4.
2. Draw a 2-manifold that has a genus of 2.
3. The object in Figure 1.1 contains 8 surfaces but is not a 2-manifold. If you were to store it in a 3D primitive in GML, which one would you choose?
4. How many interior shells does the solid in Figure 1.8 have?
5. In which direction points the normal of the interior shell of the solid in Figure 1.5?

1.6 Notes and comments

The title "*Are your polyhedra the same as my polyhedra?*" is taken from the (excellent) paper from Grünbaum (2003).

The 2D GIS data structures that can be used for storing 2-manifolds are, for instance, the half-edge (Mäntylä, 1988), the quad-edge (Guibas and Stolfi, 1985), and the doubly-connected edge list (DCEL) (Muller and Preparata, 1978); all of these store the edge of a polyhedron as the atom, with links to its adjacent edges and incident faces.

For details how the validation of a the 3D primitives can be implemented, see Ledoux (2013) and Ledoux (2018).

The official specifications documents are the following:

- ISO 19107 document: ISO (2003)
- Simple Features document: OGC (2006)
- GML specifications: OGC (2007)
- CityGML specifications: OGC (2012)

Bibliography

Grünbaum, B. (2003). Are your polyhedra the same as my polyhedra? *Discrete and Computational Geometry: The Goodman-Pollack Festschrift*. Ed. by B. Aronov, S. Basu, J. Pach, and M. Sharir. Springer, pp. 461–488.

Guibas, L. J. and J. Stolfi (1985). Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics* 4.2, pp. 74–123.

ISO (2003). ISO 19107:2003: Geographic information—Spatial schema. International Organization for Standardization.

Ledoux, H. (2013). On the validation of solids represented with the international standards for geographic information. *Computer-Aided Civil and Infrastructure Engineering* 28.9, pp. 693–706.

Ledoux, H. (2018). val3dity: validation of 3D GIS primitives according to the international standards. *Open Geospatial Data, Software and Standards* 3.1, p. 1.

Mäntylä, M. (1988). *An introduction to solid modeling*. New York, USA: Computer Science Press.

Muller, D. E. and F. P. Preparata (1978). Finding the intersection of two convex polyhedra. *Theoretical Computer Science* 7, pp. 217–236.

OGC (2006). OpenGIS Implementation Specification for Geographic information—Simple feature access. Open Geospatial Consortium inc. Document 06-103r3.

OGC (2007). Geography Markup Language (GML) Encoding Standard. Open Geospatial Consortium inc. Document 07-036, version 3.2.1.

OGC (2012). *OGC City Geography Markup Language (CityGML) Encoding Standard. Version 2.0.0*. Open Geospatial Consortium.