
# PostGIS meets the Third Dimension

**Olivier COURTIN**

**FOSS4G 2010  8 September - Barcelona**



# Presentation Plan

1) Oslandia Short Presentation

2) 3D in GIS, what for ?

3) Spatial databases standards and 3D

4) PostGIS 3D implementation

5) 3D open issues

6) Roadmap and Conclusions

OSLANDIA

# Presentation Plan

**1) Oslandia Short Presentation**

2) 3D in GIS, what for ?

3) Spatial databases standards and 3D

4) PostGIS 3D implementation

5) 3D open issues

6) Roadmap and Conclusions

OSLANDIA

# Oslandia

Young French SME **Open Source GIS company**

PostGIS Experts: Vincent Picavet and Olivier Courtin

Mainly Focuses on:

- **Spatial Databases** (PostGIS, SpatiaLite)
- OGC, ISO, INSPIRE **Standards** and **SDI architecture**
- **Complex analysis**: Routing, Network and Graphs Solutions

Oslandia Ecosystem:

OGC® Member

OSGeo Your Open Source Compass

OSLANDIA

# Oslandia's Technologies

3D GDAL GEOS

GRASS GraphServer INSPIRE MapServer

OGC PgRouting **PostGIS**

PostgreSQL Spatialite TinyOWS

TileCache PyWPS QGIS

OSLANDIA

# Oslandia, Find us on FOSS4G

Running Long and Complexes Processes with PostGIS

*Vincent Picavet: Wednesday - 12h00 – Sala 6*

PostGIS Meets Third Dimension

*Olivier Courtin : Wednesday - 12h30 – Sala 6*

State of the Art of FOSS4G for Topology and Network Analysis

*Vincent Picavet: Thursday – 14h30 – Sala 5*


Oslandia is Bronze Sponsor

Breakout Session: Spatial Database

Code Sprint on Friday: PostGIS

# Presentation Plan

1) Oslandia Short Presentation

**2) 3D in GIS, what for ?**

3) Spatial databases standards and 3D

4) PostGIS 3D implementation

5) 3D open issues

6) Roadmap and Conclusions

OSLANDIA

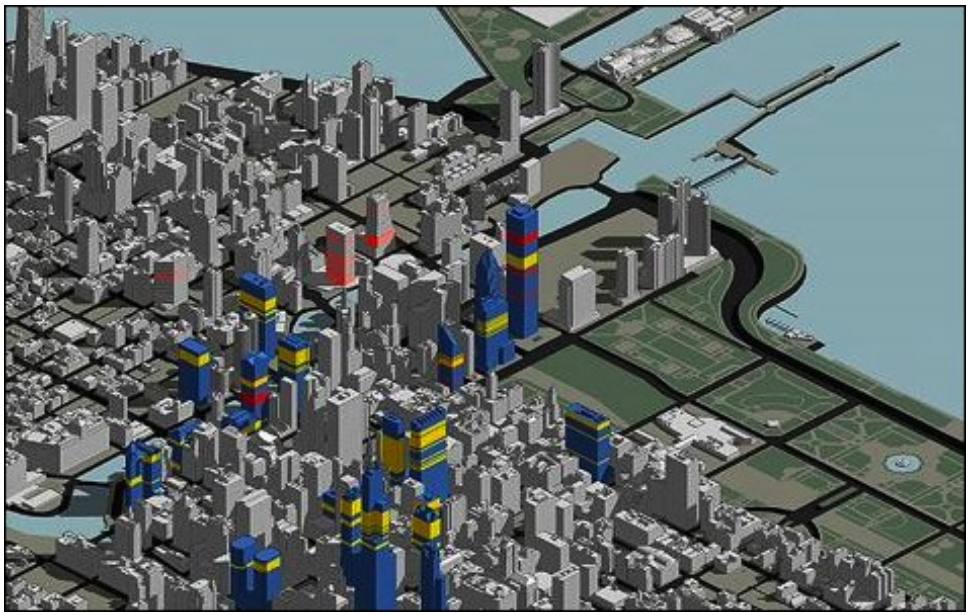
# 3D GIS: A meeting point

**BIM:**

Focus on **Building** model

CAD/CAO world

**IFC** standard



**CIM:**

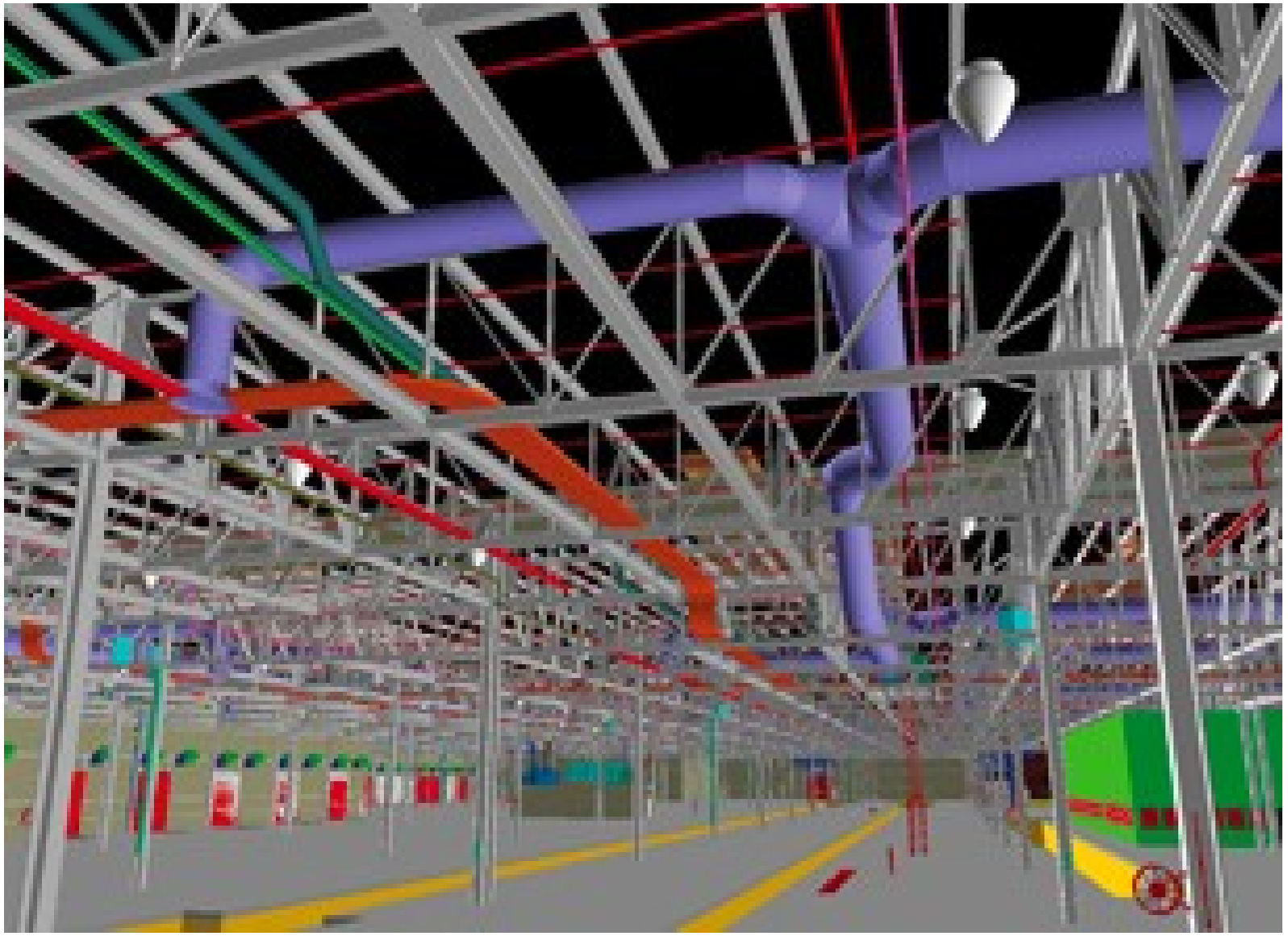
Focus on **City** model

GIS world

**CityGML** standard

OSLANDIA

# 3D BIM: Facilities and Physics Networks



OSLANDIA

# 3D CIM: City Model

OSLANDIA

# CityGML Overview

- OGC Standard
- 3D format XML based
- Use GML 3.1.1 for geometries encoding
- INSPIRE Recommandation

Mostly **focus on CIM** (rather than BIM)

More an **interoperability GIS format** exchange (rather than direct 3D rendering)

# CityGML Supports

- Geometries and attributes handling
- Textures
- Extensible Application Model (ADE)
- Level Of Details (LOD)

# CityGML: (LOD) Levels Of Details

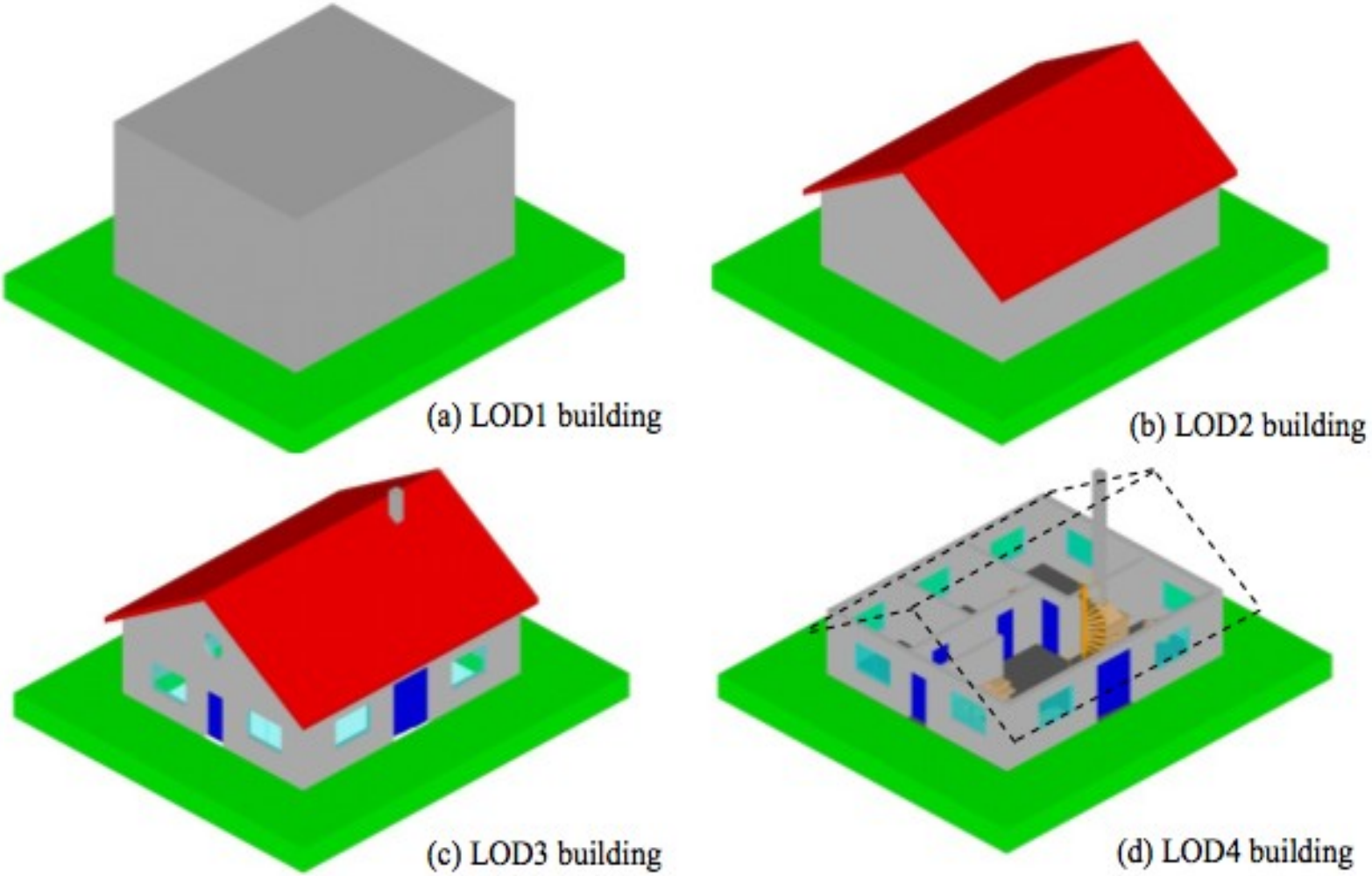



Fig. 27: Building model in LOD1 – LOD4 (source: Research Center Karlsruhe).

OSLANDIA

# Presentation Plan

1) Oslandia Short Presentation

2) 3D in GIS, what for ?

**3) Spatial databases standards and 3D**

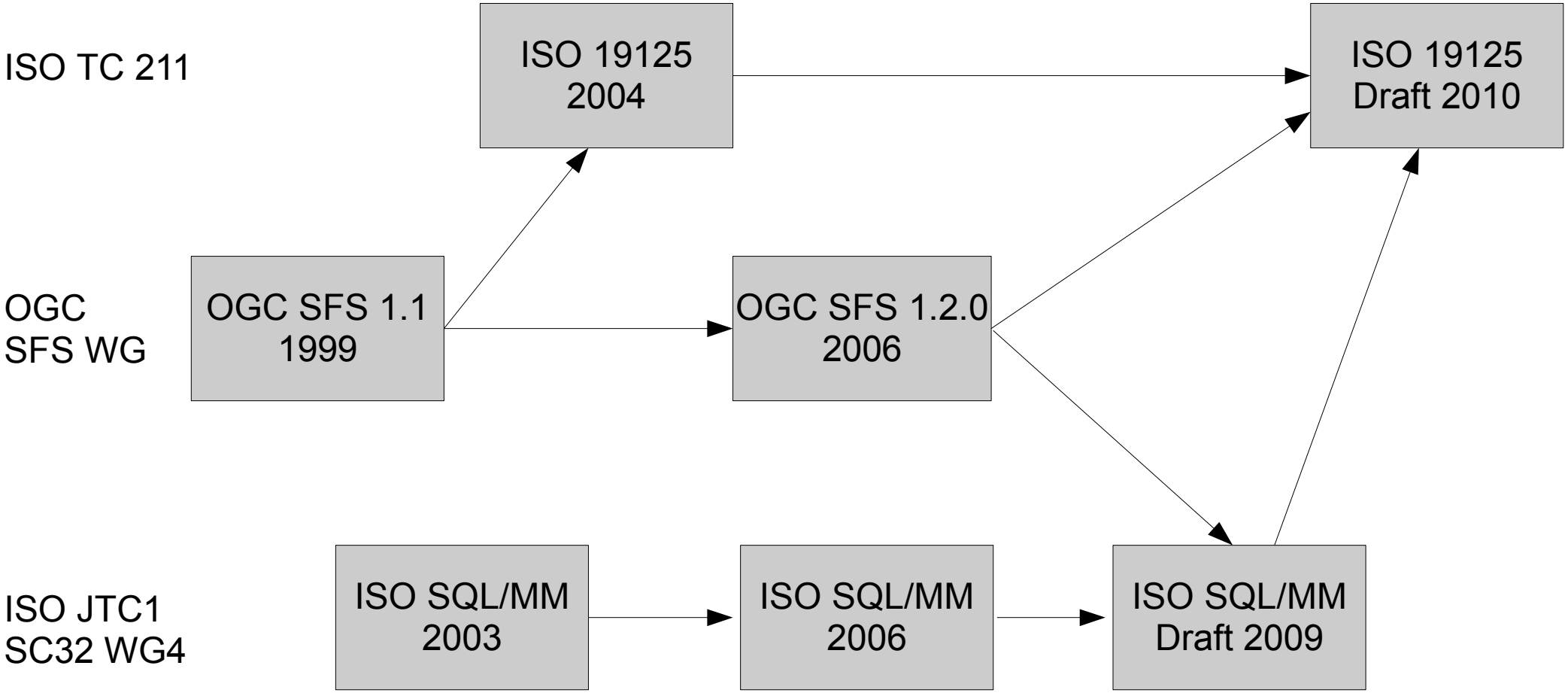
4) PostGIS 3D implementation

5) 3D open issues

6) Roadmap and Conclusions

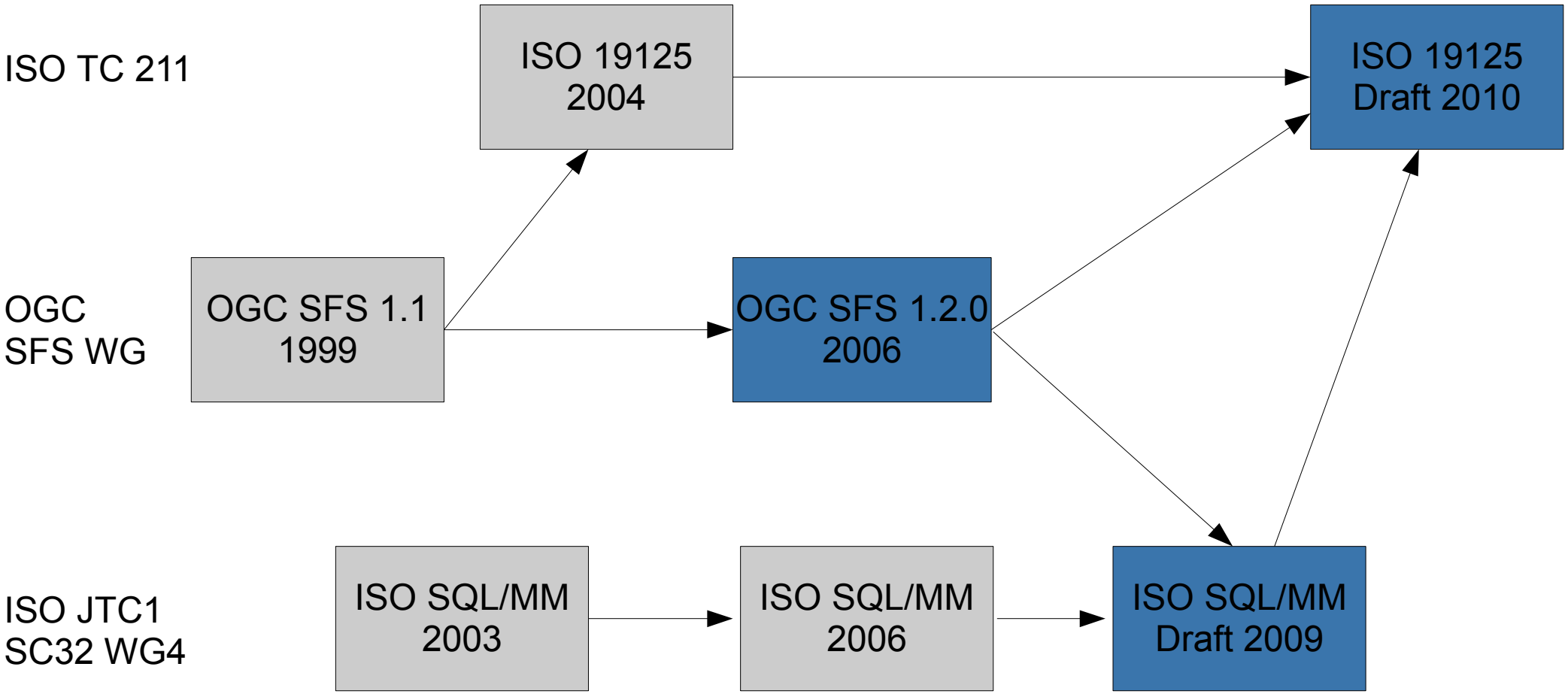
OSLANDIA

# Spatial database standards




OSLANDIA

# Spatial database standards: 3D concepts




OSLANDIA

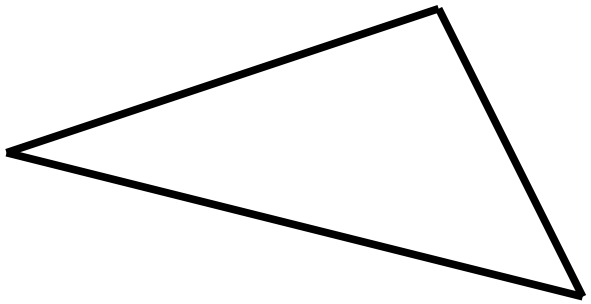
# New Surface types: Triangle

**One exterior ring** with 3 differents points
(and 1 point more to close the ring)

No interior ring (i.e **no hole**)

**Points** must **not** be **colinears**

Triangle could be 2D, 3D, 3DM or even 4D



```
TRIANGLE((0 2, 10 4, 12 0, 0 2))
```

OSLANDIA

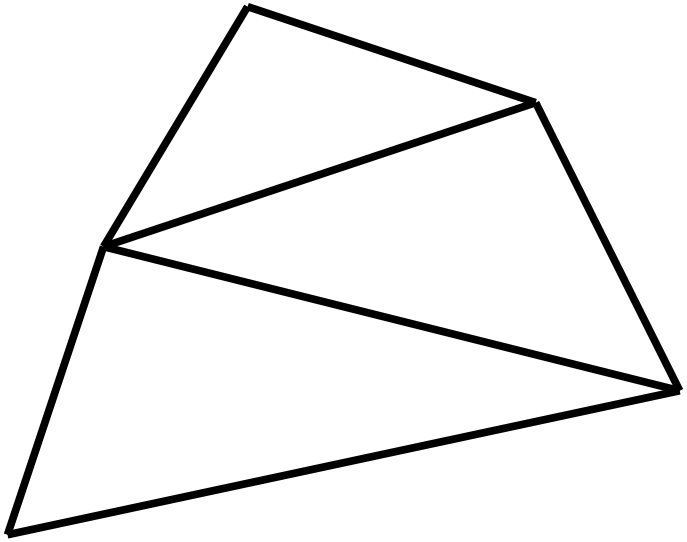
# New Surface types: TIN

Collection of **triangles connected by edges**

Every triangle share **same orientation**

TIN **could enclose a solid** (or not)

TIN could be 2D, 3D, 3DM or even 4D



```
TIN(((0 2, 10 4, 12 0, 0 2)),
    ((0 2, -2 -6, 12 0, 0 2)),
    ((0 2, 10 4, 5 8, 0 2)))
```

OSLANDIA

# New Surface types: PolyhedralSurface

Collection of **polygons connected by edges**

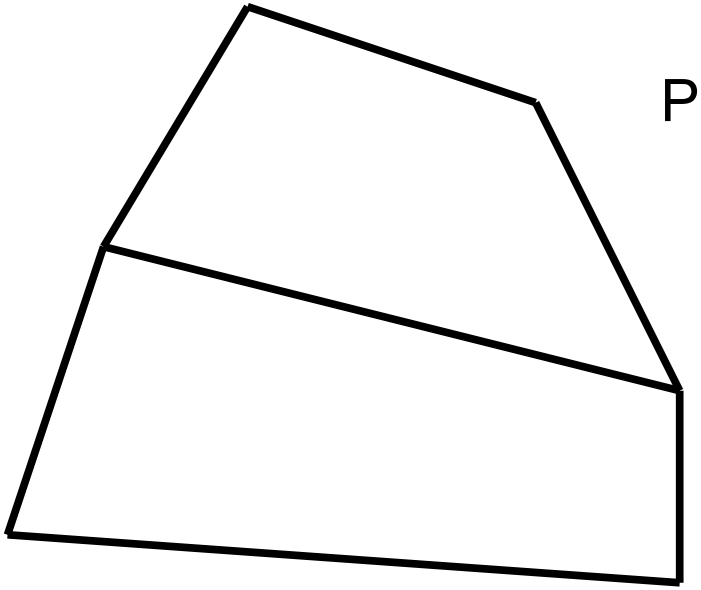
Every polygon share **same orientation**

Points of the polygon must be **coplanar** (enough)

Polygons could have **internal rings** (i.e holes)

PolyhedralSurface **could enclose a solid** (or not)

PolyhedralSurface could be 2D, 3D, 3DM or even 4D

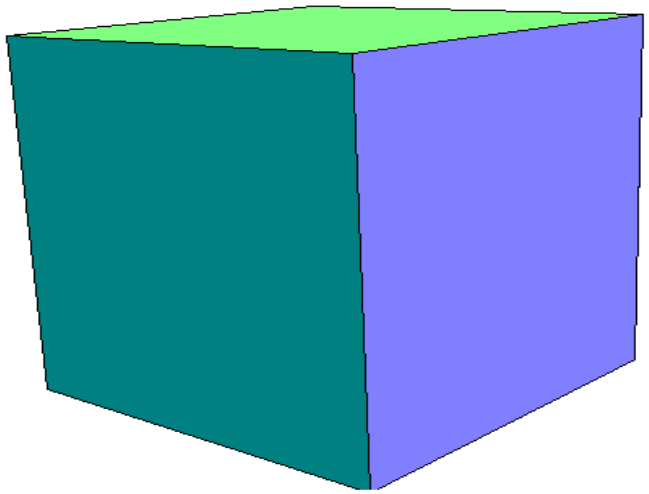


```
POLYHEDRALSURFACE(
 ((0 2, 10 4, 12 0, 5 8, 0 2)),
 ((0 2, -2 -6, 12 -6, 12 0, 0 2)))
```

OSLANDIA

# New Surface types: PolyhedralSurface

A 3D **PolyhedralSurface** example, **enclosing a cube**



```
POLYHEDRALSURFACE(
  ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
  ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),
  ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
  ((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
  ((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)),
  ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)))
```

OSLANDIA

# Spatial database standards: 3D specs

| | OGC SFS 1.2 2006 | Draft ISO SQL/MM 2009 | Draft ISO 19125 2010 |
|---|---|---|---|
| Triangle | **No** | Yes | Yes |
| TIN | Yes | Yes | Yes |
| PolyhedralSurface | Yes | Yes | Yes |
| Functions on TIN and PolyhedralSurface handling | Yes | Yes | Yes |
| Functions on 3D Topology and measures (Distance, Intersects...) | **No** | Yes | Yes |
| Vertical Datum | **No** | **No** | Yes |

# Presentation Plan

1) Oslandia Short Presentation

2) 3D in GIS, what for ?

3) Spatial databases standards and 3D

**4) PostGIS 3D implementation**

5) 3D open issues

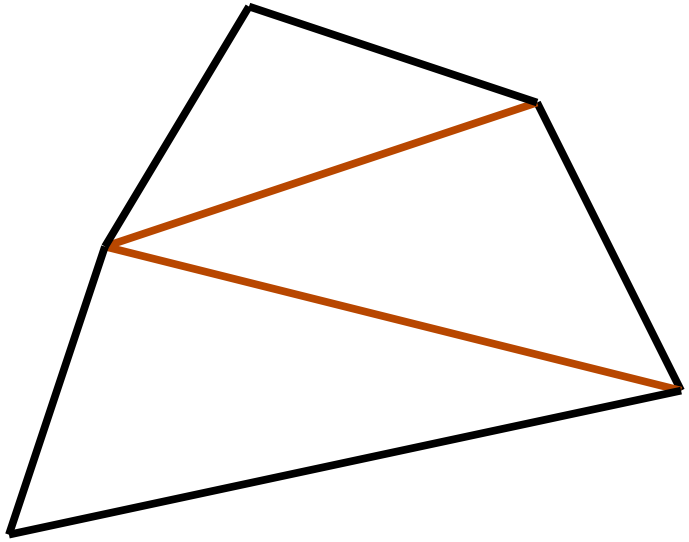
6) Roadmap and Conclusions

OSLANDIA

# Spaghetti storage model is not enough

On **common PostGIS geometry** storage, geometry **spaghetti model is used**.

On connected surfaces it leads to **redundant informations** (red edges below) (and also to possible topology artefacts)



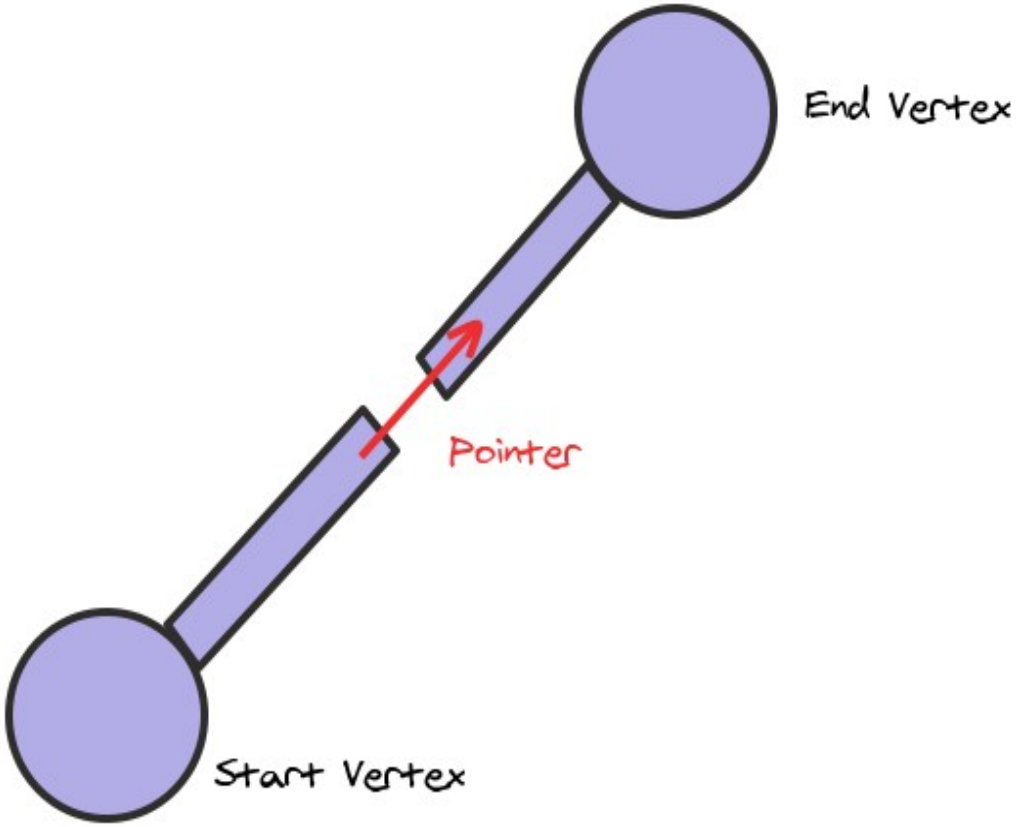
Aim for connected surfaces is to **store topology** geometry based on edges and faces

Aim is also to know if a geometry **is wheter a solid or not** (without additional computation)

OSLANDIA

# HowTo Store: Using Half Edges

**Best structure** to store **PolyhedralSurface and TIN** topology **is** based on **Half Edges**

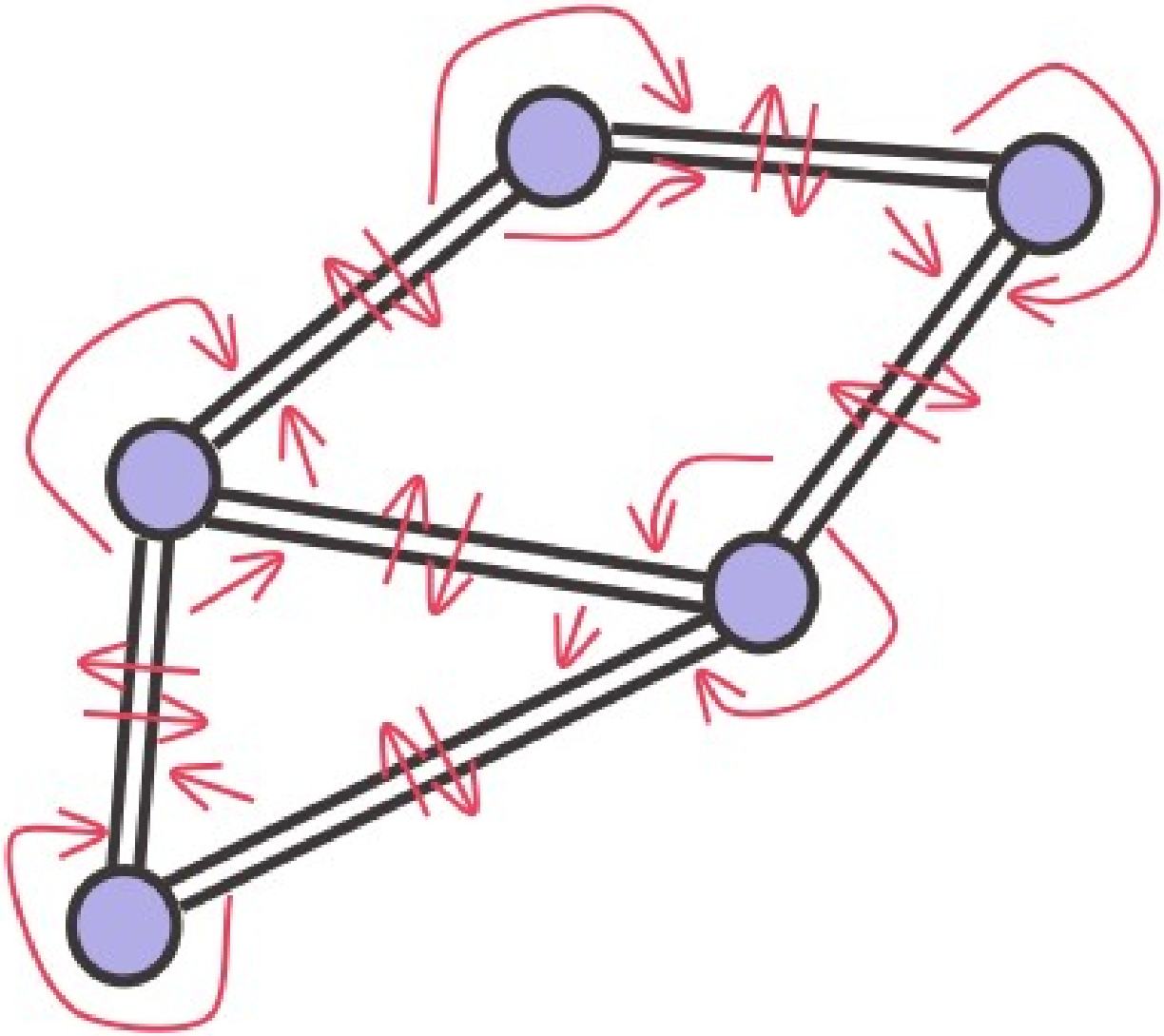



**No vertex stored twice**

**Edge Orientation** is **given by** the **pointer** between each pair of half edges.

OSLANDIA

# HowTo Store: Double Connected Edge List



A Double Connected Edge List (**DECL**)

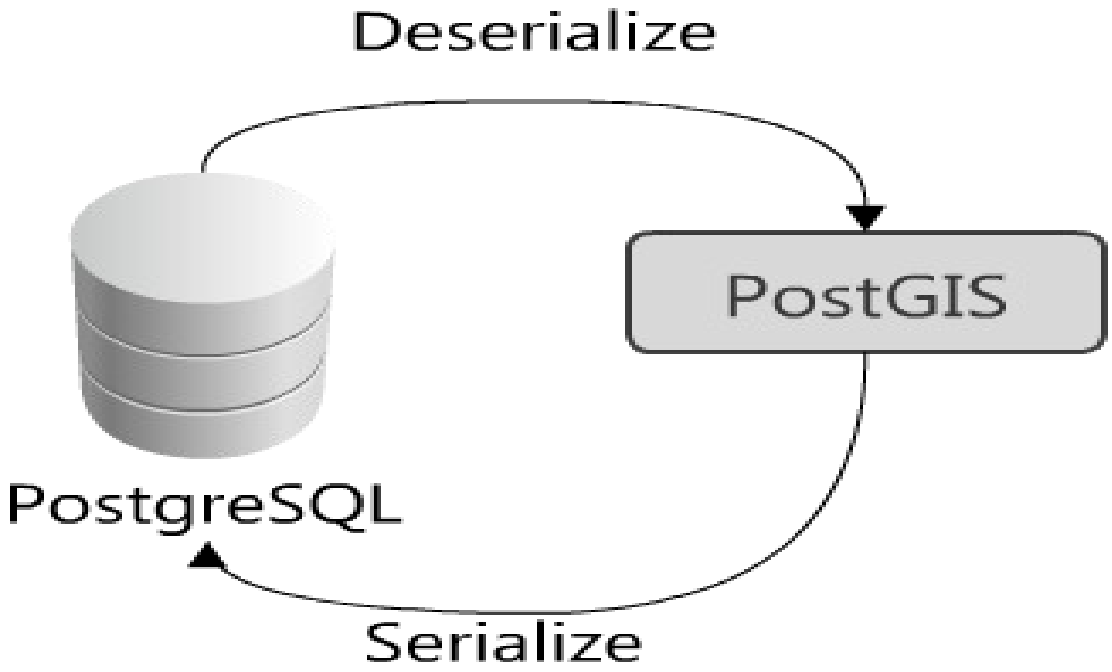
**Each arrow** means a **pointer**

*Structure used by CGAL and OpenMeshes*

OSLANDIA

# Handle PostGIS Serialization

**PostGIS use** (de)**serialize** mecanism **to store** data **into PostgreSQL**




But **serialization** of a **DCEL** is **not efficient** at all !

So we use indexed array to store edges
(implies a limit to ~4 billions of vertex per feature)

OSLANDIA

# Face and Edge Arrays Structure I

```
typedef struct
{
        POINT4D *s;             /* Edge starting point */
        POINT4D *e;             /* Edge ending point */
        int count;              /* Count how many time this edge is used in the TGEOM.
                                   Caution: We don't care about edge orientation ! */
} TEDGE;

typedef struct
{
        int nedges;
        int maxedges;
        int *edges;             /* Array of edge index, a negative value
                                   means that the edge is reversed */
        int nrings;
        POINTARRAY **rings;     /* Internal rings array */
} TFACE;
```

OSLANDIA

# Face and Edge Arrays Structure II

```
typedef struct
{
        uchar type;
        uchar flags;
        uint32 srid;            /* 0 == unknown */
        BOX3D *bbox;            /* NULL == unneeded */
        int nedges;
        int maxedges;
        TEDGE **edges;
        int nfaces;
        int maxfaces;
        TFACE **faces;
} TGEOM;
```

OSLANDIA

# Compliant Functions Availables on Trunk

Box2D
Box3D
GeometryType
ST_Affine
ST_Area
ST_AsBinary
ST_AsEWKT
ST_AsEWKB
ST_AsGML
ST_Dimension
ST_Dump
ST_DumpPoints
ST_Expand
ST_Extent
ST_Extent3D
ST_FlipCoordinates
ST_Force_2D
ST_Force_3D
ST_Force_3DZ
ST_GeomFromEWKT
ST_GeomFromEWKB
ST_GeomFromGML
ST_GeometryN
ST_GeometryType
ST_IsClosed
ST_MemSize
ST_NumGeometries
ST_NumPatches
ST_PatchN
ST_Perimeter
ST_Perimeter3D
ST_Rotate
ST_RotateX
ST_RotateY
ST_RotateZ
ST_Scale
ST_Shift_Longitude
ST_Transform

# Presentation Plan

1) Oslandia Short Presentation

2) 3D in GIS, what for ?

3) Spatial databases standards and 3D

4) PostGIS 3D implementation

**5) 3D open issues**

6) Roadmap and Conclusions

OSLANDIA

# Open Issues Lists

1) Triangulation
2) TIN Simplification
3) CityGML Loader
4) IsValid geometries check
5) Vertical Datum
6) Multidimensionnal Index
7) TIN for DEM Storage
8) Texture handling
9) Google Earth I/O
10) 3D Topology functions

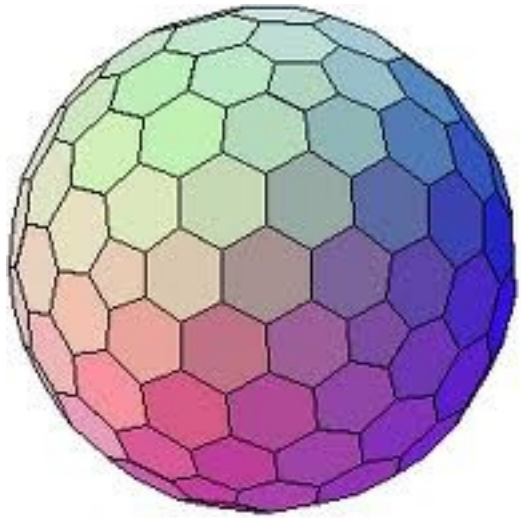
OSLANDIA

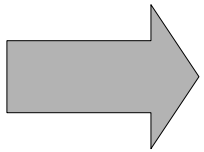
# Open Issue #1: Triangulation

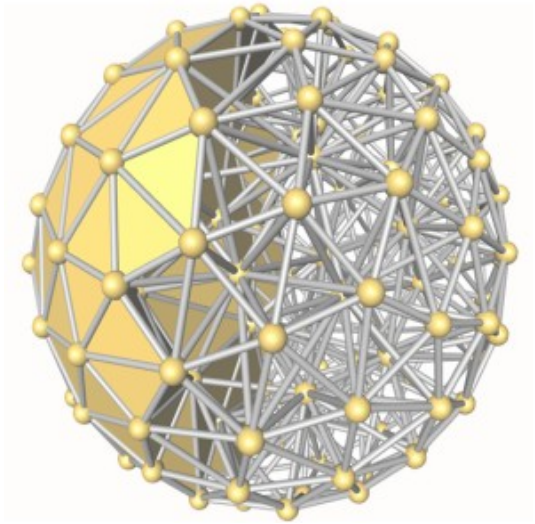
Lot of **3D usages** only deal with **TIN** and not with PolyhedralSurface

So we must be able to **convert**
**PolyhedralSurface to TIN**

*Known Implementations: CGAL, TetGen*







OSLANDIA

# Open Issue #2: Tin Simplification

**Reduce** the number of **triangles**

**Preserve** the **volume**

**Preserve** the global and local **shapes**

*Source: CGAL*

Fast and memory efficient Algorithm:
**Lindstrom and Turk**

*Known Implementations: CGAL, GTS*

OSLANDIA

# Open Issue #3: CityGML Loader

**GML 3 handles** a lots of **geometry types** that **spatial database standard don't**. (true solid...)

**GML allow composition** of several solids into a single feature

Interesting to be able to **downsize LOD** *(e.g: LOD 3 -> LOD 2)*

CityGML extension application schema (**ADE**)

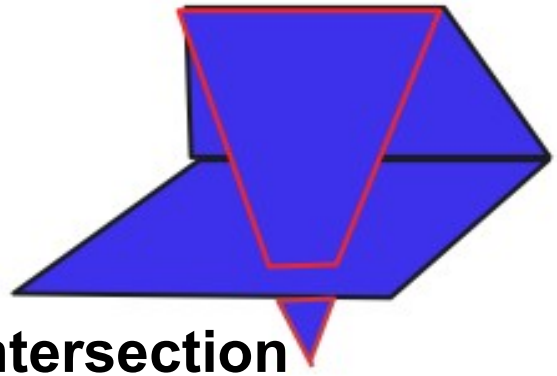
Implies **Triangulation** and **simplification** to import TIN into database and **ST_Union on 3D**

OSLANDIA

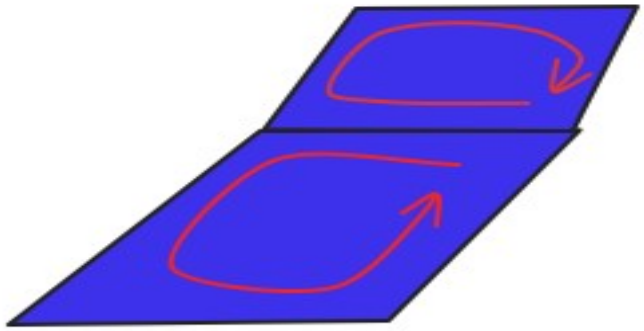
# Open Issue #4 IsValid Check

**IsValid checks** are currently done by **GEOS**

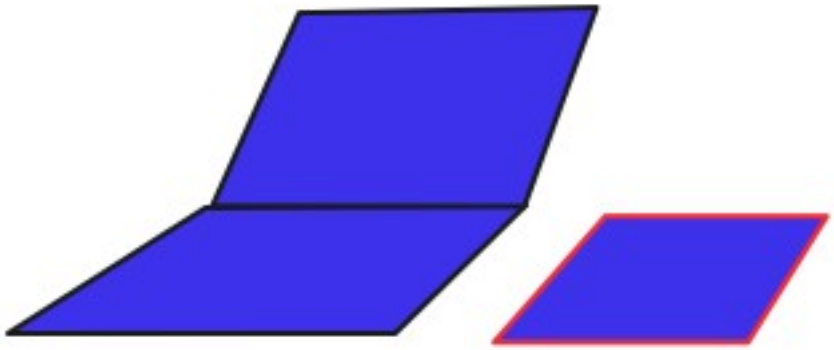
GEOS **don't care** about Z dimension nor **3D types**



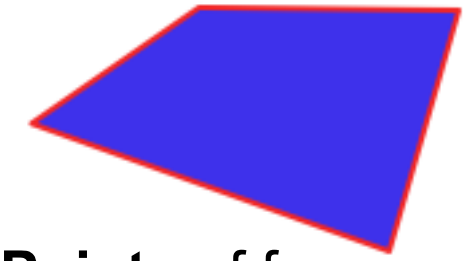
No **Self Intersection**



All Faces **Well Oriented**



All **Faces** edges **connected**

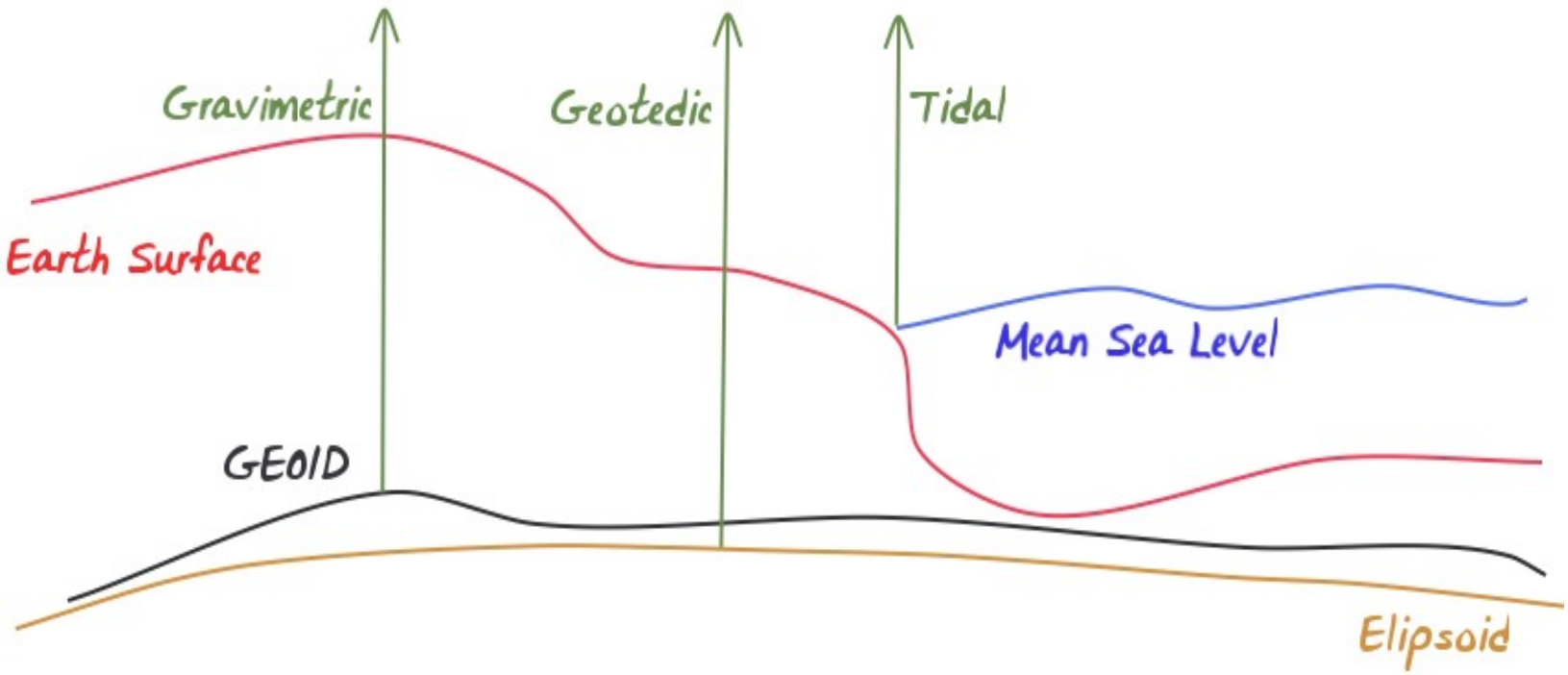


**Points** of face
are **coplanars**

OSLANDIA

# Open Issue #5: Vertical Datum

**SRID** is used to reference a Coordinate Reference System in 2D

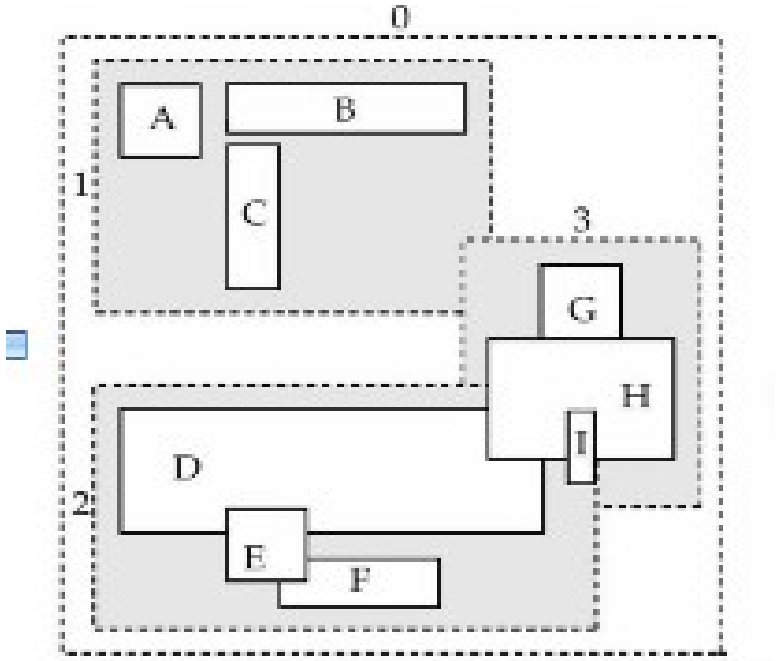



We need also to be consistent add a **vertical SRID** for Z axis

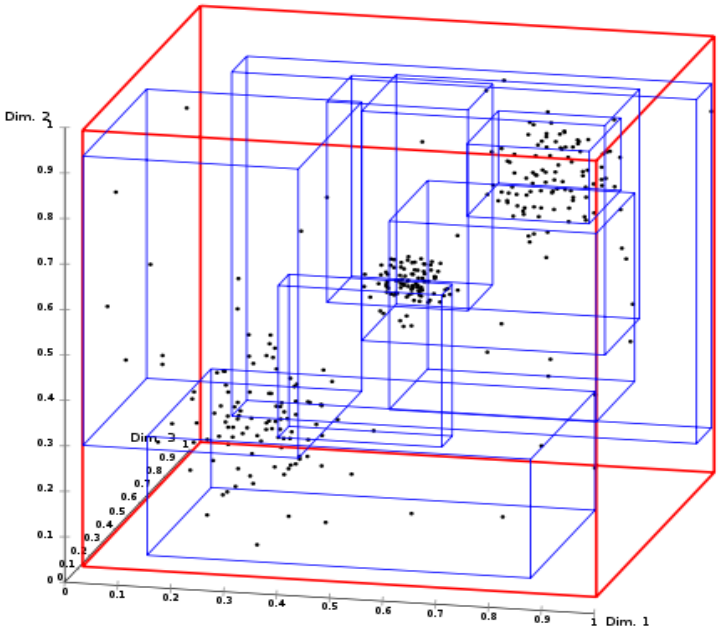
**Proj4** begins to add **vertical datum support**
*(cf Franck's conference yesterday)*

OSLANDIA

# Open Issue #6: Multidimensional Index




2D Rtree Index

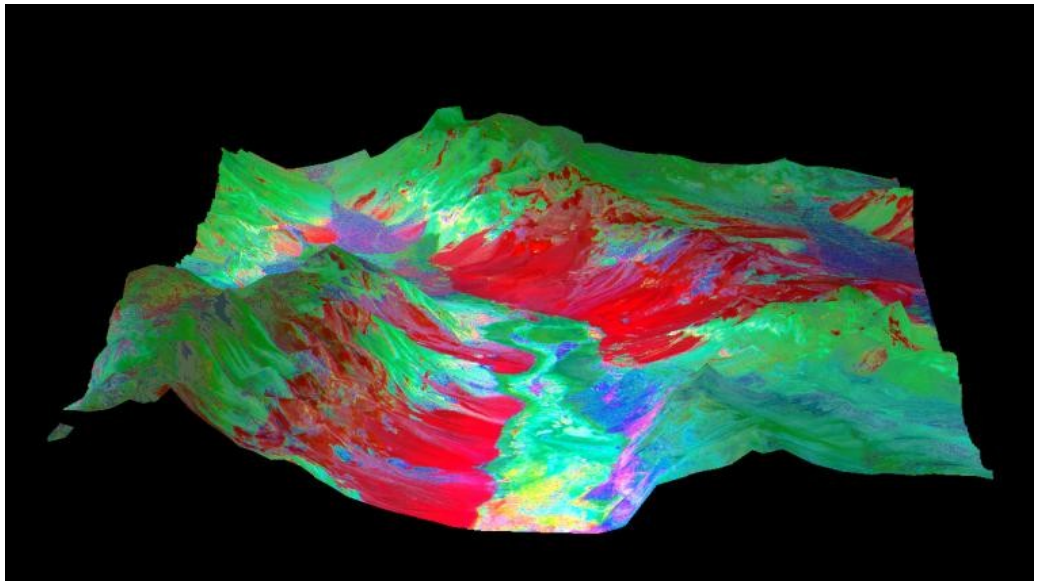



3D Rtree Index

**Multidimensional index** implementation will be needed to **improve** again **performances** on 3D data.

OSLANDIA

# Open Issues #7 DEM storage



Source: NASA

Terrain DEM model use also often TIN structure

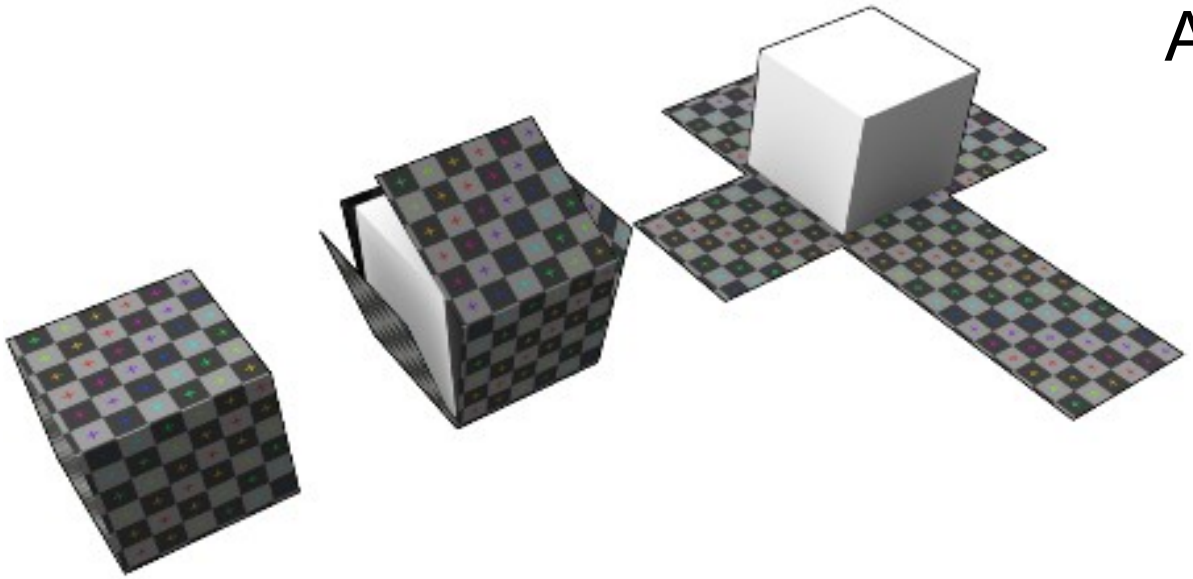
But it's a **NONSENSE** to store a whole DEM in a single PostGIS TIN feature !

To store efficiently:

- Ability to **split the** whole **TIN** into smaller pieces
- Store each pieces in a different PostgreSQL row
- 2D spatial index to access quickly to each piece

OSLANDIA

# Open Issue #8: Texture



A Texture is composed of:

A 2D geometry (UV)

Associated to an image

Explore the ability to deal with **texture handling** throught **WKT Raster**

OSLANDIA

# Open Issue #9: Google Earth I/O

For 3D **Google Earth use Collada** embedded inside KML (Model tag)

GE only deals with **TIN smaller** than **21845 triangles**

KML **altitude** could be **relative to ground or absolute** (geocentric)

**COLLADA** could **use texture** images

Google Earth input/output handle implies: ***triangulation***, ***TIN simplification***, ***vertical datum*** *and* ***texture*** *support.*

OSLANDIA

# Open Issue #10: Topology Operations

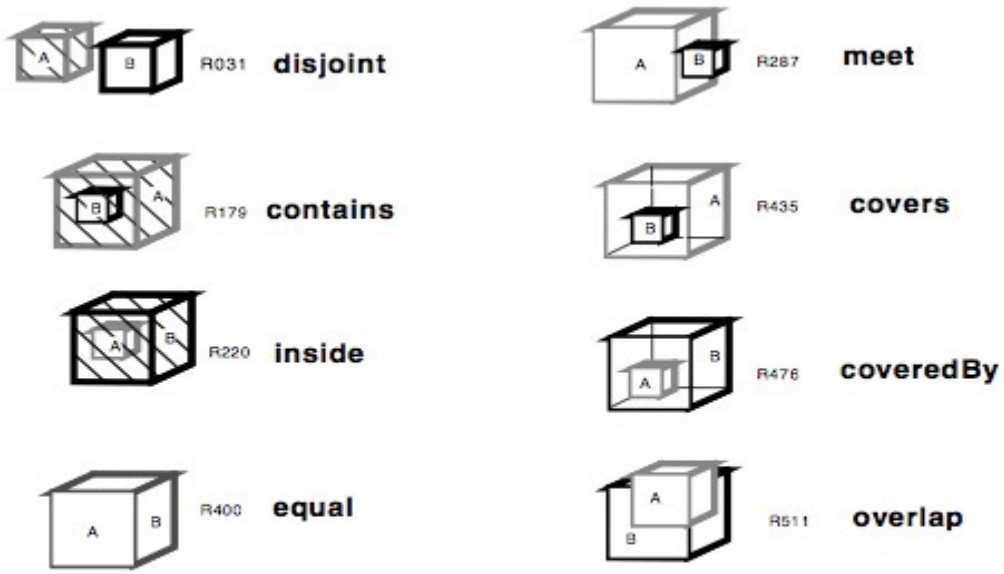



Figure 8: The 9-intersection model: possible relationships between 3D and 3D objects
(source Zlatanova, Rahman and Shi 2002)

Now **Topology operations** are done by **GEOS**

**GEOS** (and JTS) **only** handle **2D** (OGC SFS 1.1)

**3D Topology Operations** are in latest SQL/MM and ISO 19125 **drafts**

There's a need for a robust **C++ Topology lib** able to fully handle **ISO 19107** (2D and 3D).

OSLANDIA

# Presentation Plan

1) Oslandia Short Presentation

2) 3D in GIS, what for ?

3) Spatial databases standards and 3D

4) PostGIS 3D implementation

5) 3D open issues

**6) Roadmap and Conclusions**

OSLANDIA

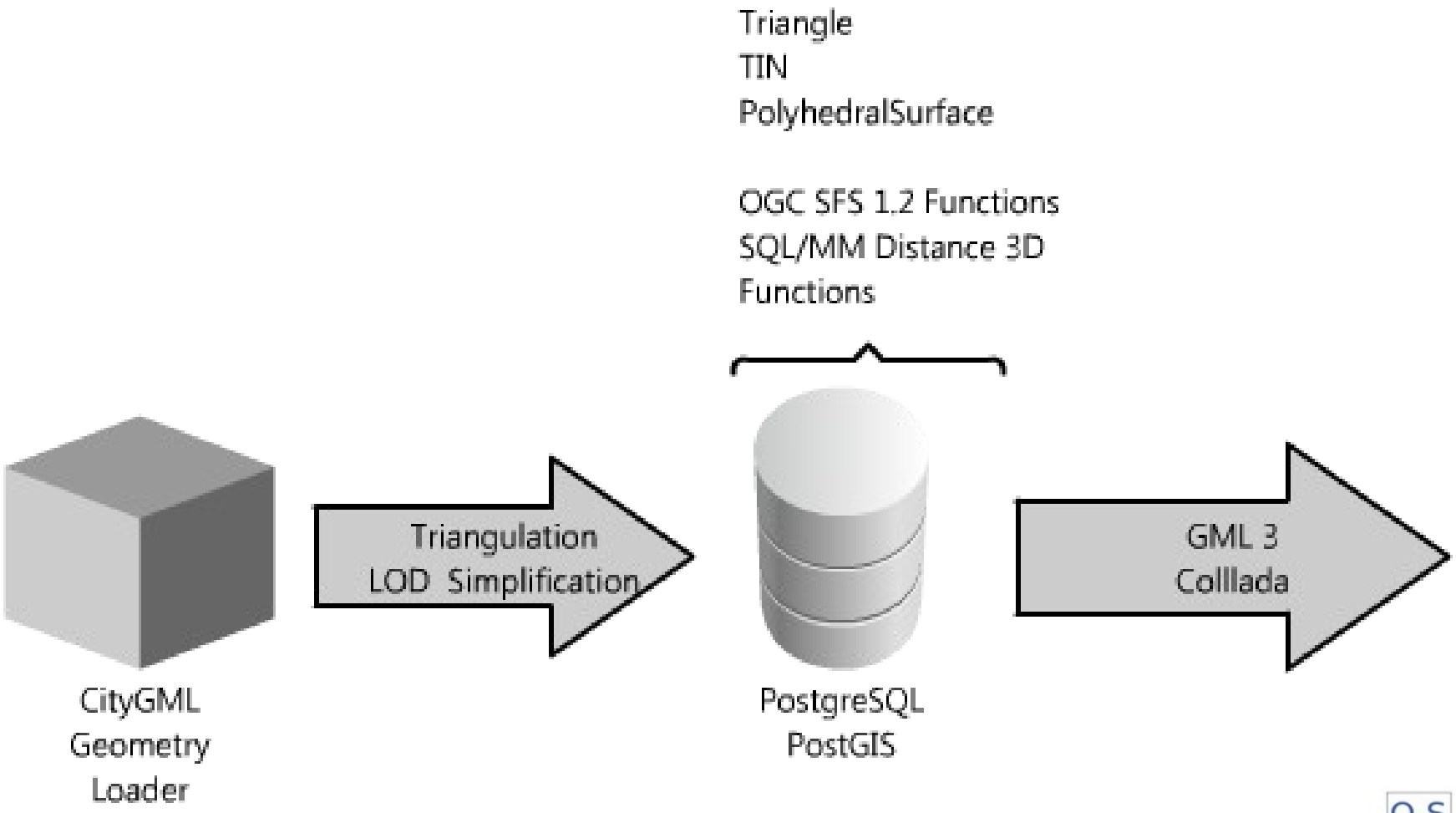
# Who contributes to PostGIS 3D ?


**Regina Obe** — Quality assurance and documentation

**Nicklas Aven** — 3D Distance functions


**Olivier Courtin** — PostGIS 3D geometry type and import/export functions

# 3D Roadmap - PostGIS 2.0




OSLANDIA

# And Then What's Next ?

Well **it depends on you** too !

We are looking for help:

**C/C++ Developpers** to tackle some open issues

**Fund** to finance development effort

# Conclusions

**We** already **implement** in PostGIS trunk the only published **standard** database available **for 3D** (OGC SFS 1.2)

So, **PostGIS 2.0** will be able to **deal with 3D geometries** primitives and some related additional spatial functions

OSLANDIA

# Conclusions

**We** already **implement** in PostGIS trunk the only published **standard** database available **for 3D** (OGC SFS 1.2)

So, **PostGIS 2.0** will be able to **deal with 3D geometries** primitives and some related additional spatial functions

A **full 3D support** (topology, vertical datum, textures...) is an huge work and **will require** related **funding** / effort.

**3D Topology library** is a **broader issue** than only PostGIS concern, and could/should be **shared by other FOSS4G** apps.

OSLANDIA

# Contacts

Olivier COURTIN
olivier.courtin@oslandia.com

**www.oslandia.com**

OSLANDIA