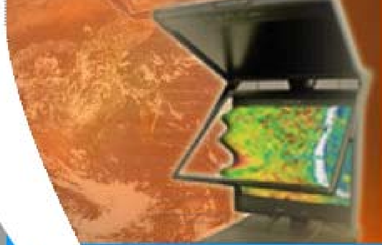


Lunchmeeting GIS Technology

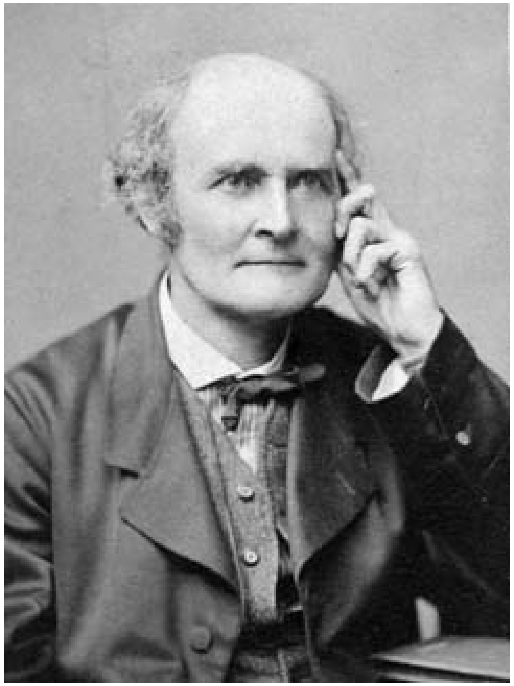
Friday September 1st 12.00-14.00h

RGI project 3D Topography

2 PhD's, 2 presentations:

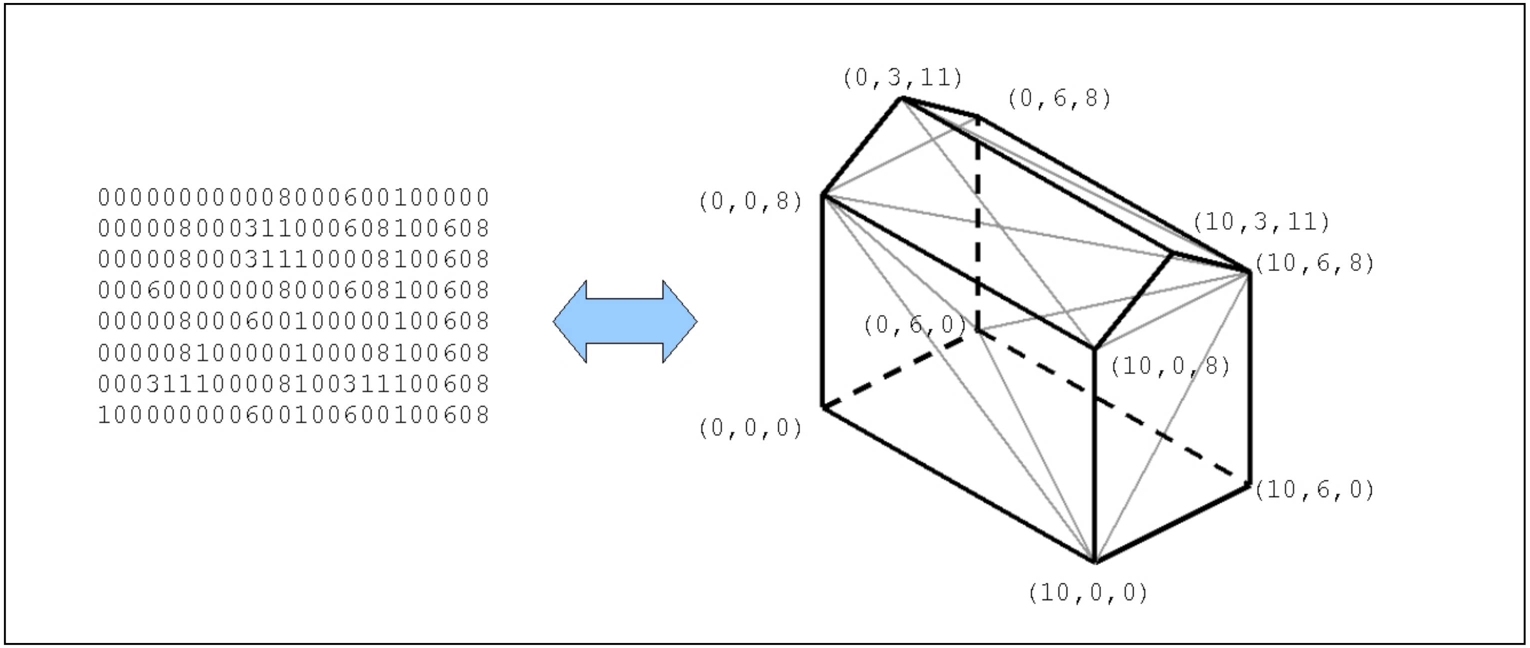
- Friso Penninga (focus: data structure)
- Sander Oude Elberink (focus: data collection)

Check www.gdmc.nl/3dtopo !

 +  + 

= 3D GIS?

Poincaré simplicial homology for 3D volume modeling



Friso Penninga
Delft University of Technology,
section GIS Technology

TU Delft

Outline

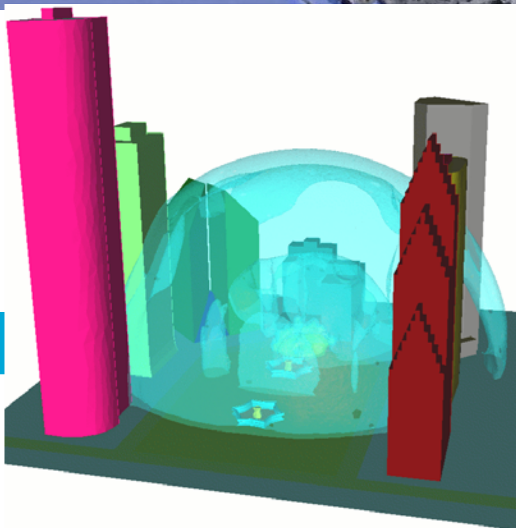
- Introduction
- PhD project history in a nutshell
- Basics of modelling approach
- Poincaré's formalism of simplicial homology
- Concept of Poincaré-based TEN structure
- Implementation → illustration of concept
- Conclusions & future research
- Discussion

Introduction:
Need for 3D Topography

- Real world consists of 3D objects
- Objects + object representations get more complex due to multiple land use

Applications in:
- Sustainable development (planning, analysis)
- Support disaster management

3D Topography: more than visualization!

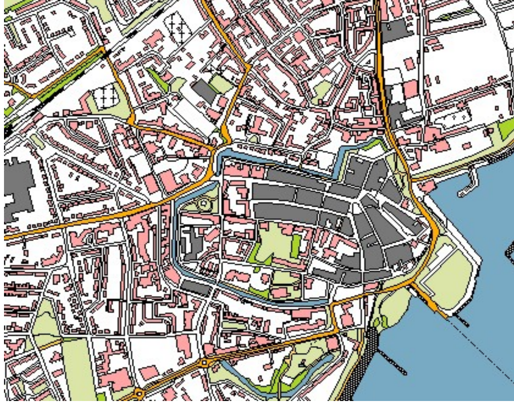


Introduction:
Research Goal

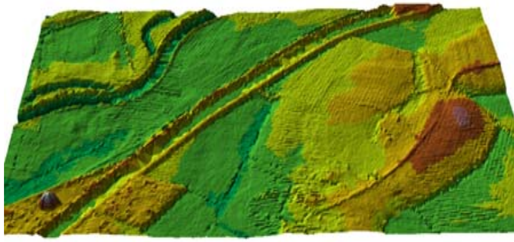
Develop a new topographic model to be realized within a robust data structure and filled with existing 2D, 2.5D and 3D data

Data structure:

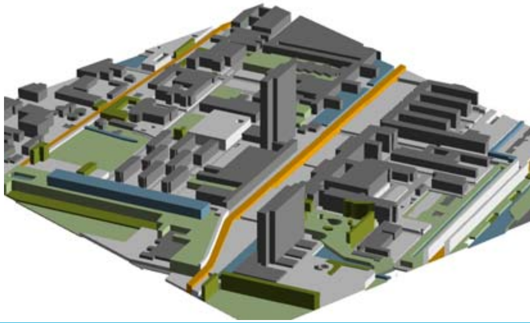
design / develop / implement a data structure that supports 3D analyses and maintains data-integrity



+



=



Outline

- Introduction
- PhD project history in a nutshell
- Basics of modelling approach
- Poincaré's formalism of simplicial homology
- Concept of Poincaré-based TEN structure
- Implementation → illustration of concept
- Conclusions & future research
- Discussion

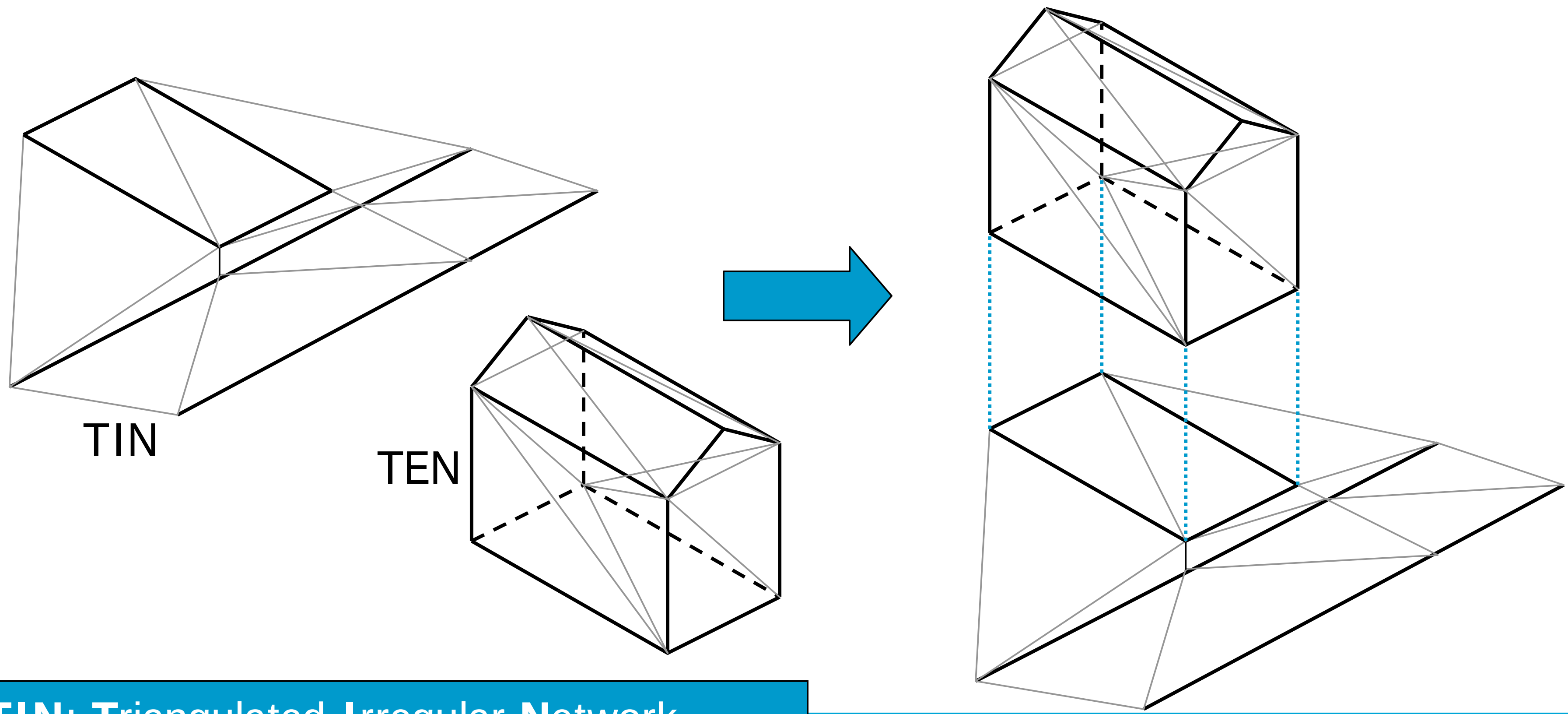
History in a nutshell:
First ideas

- July 2004: *“Realization of a three dimensional topographic terrain representation in an integrated TIN/TEN model”* (Agile 2005)

Proposed data model:

2.5D TIN + 3D TEN

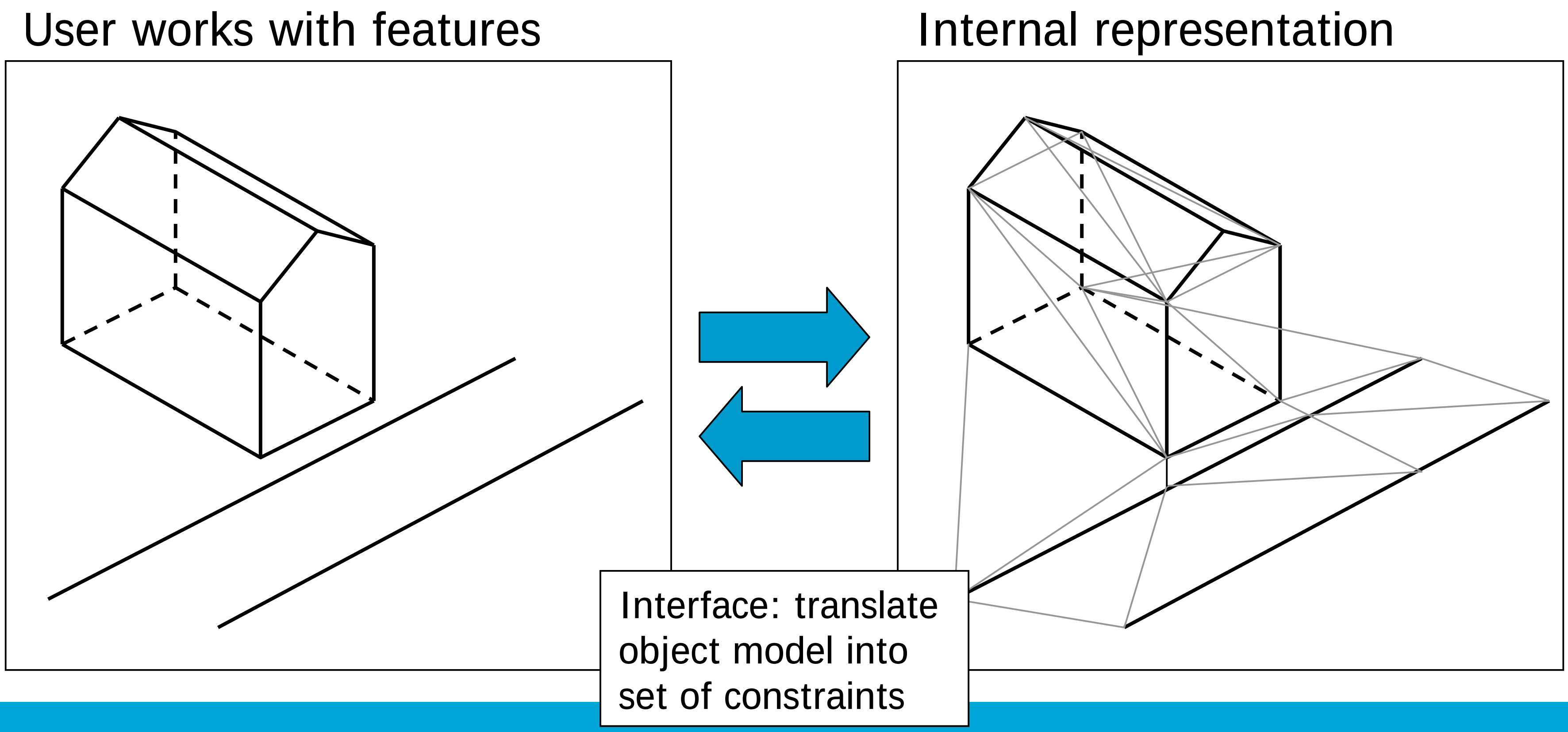
Model terrain in 2.5D, ‘glue’ 3D models on top or below 2.5D terrain



TIN: **T**riangulated **I**rregular **N**etwork

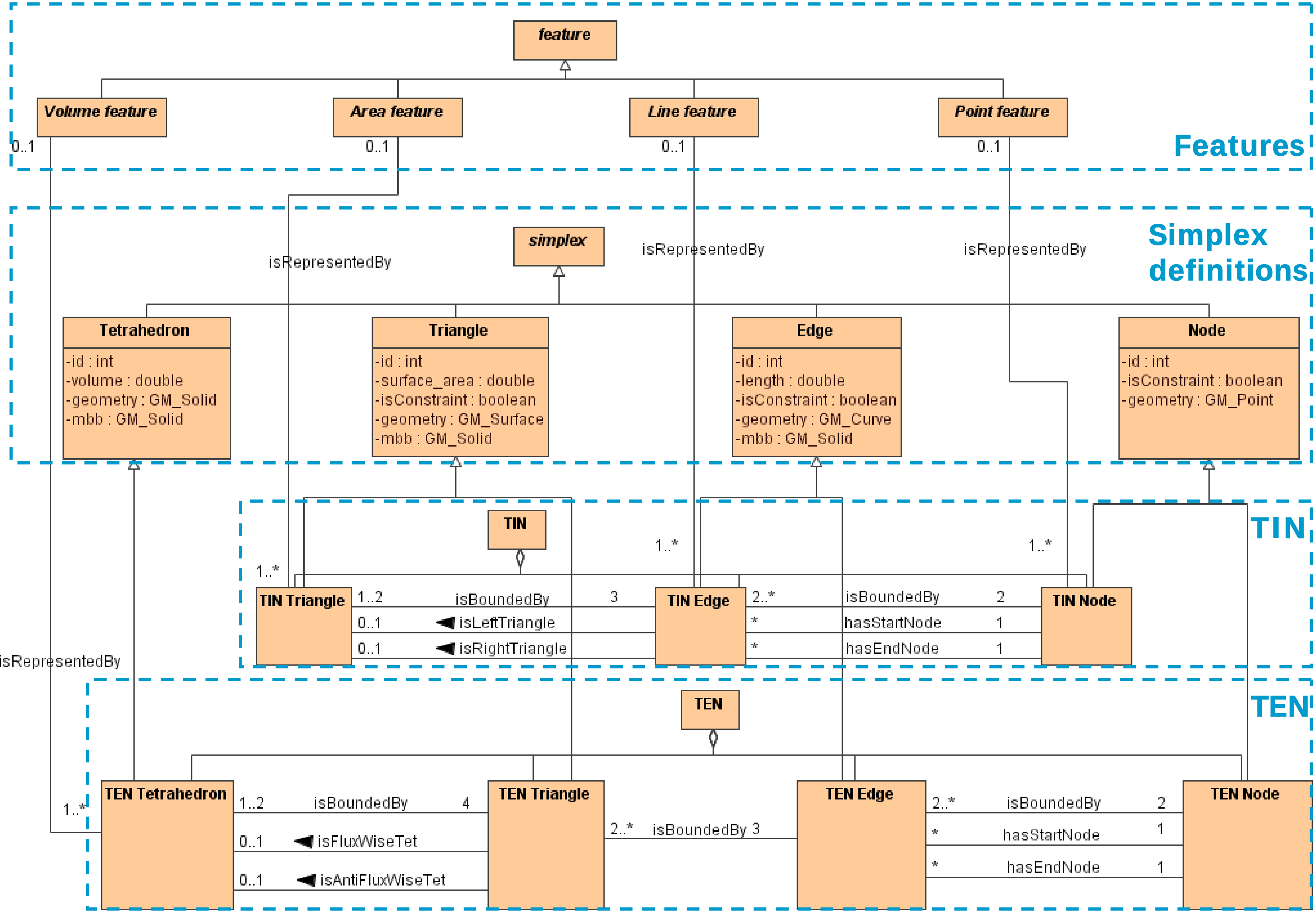
TEN: **Te**trahedronized Irregular **N**etwork

Basic principles (3/3)



Feature-based Integrated TIN/TEN model in UML

Presentation sounding board, September 24 2004



History in a nutshell:
Continuation on hybrid approach

- July 2004: *"Realization of a three dimensional topographic terrain representation in an integrated TIN/TEN model"* (Agile 2005)
- September 2004: Linking TIN and TEN model (glue, stitch & nail)

TU Delft

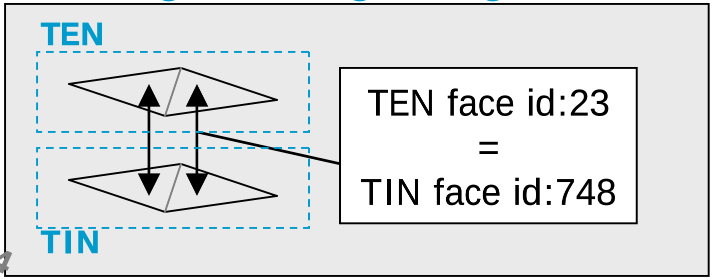
Connecting TIN+TENs
Problems

How to ensure identical faces or edges in TIN surface and TEN bottom?

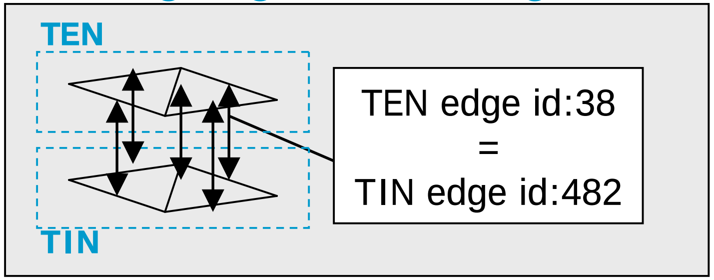
Might be problematic due to addition of Steiner points in triangulation / tetrahedronization (iterating process?)

Possible solution: addition of Steiner points in interior (common in mesh refinement, but rather unusual in GIS (nodes represent point measurents))

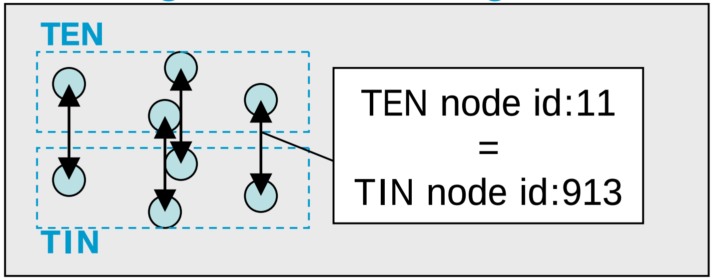
Linking faces: 'glueing'



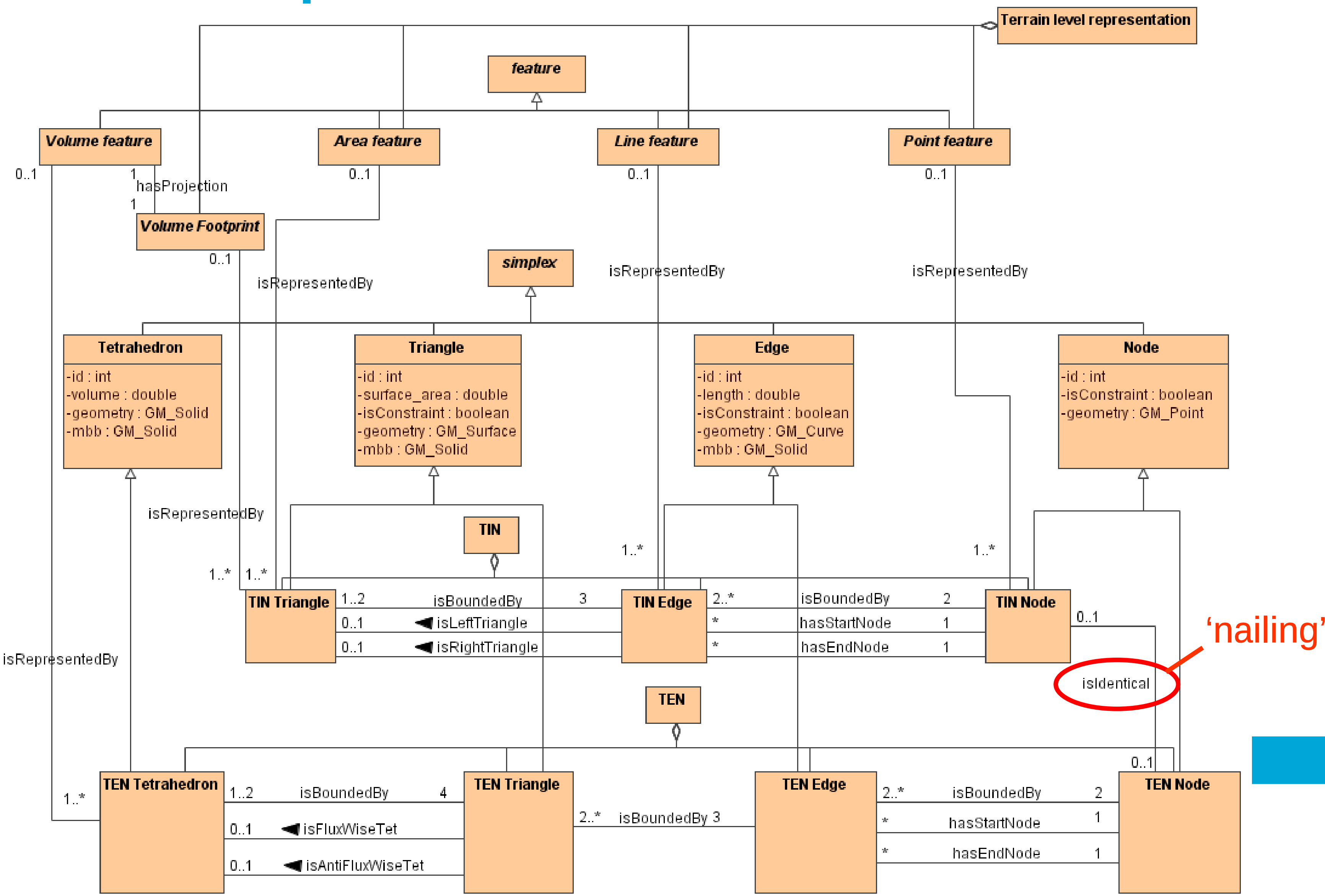
Linking edges: 'stitching'



Linking nodes: 'nailing'



Implementation:
Conceptual model



History in a nutshell:
Switch to full 3D approach

- July 2004: *"Realization of a three dimensional topographic terrain representation in an integrated TIN/TEN model"* (Agile 2005)
- September 2004: Linking TIN and TEN model (glue, stitch & nail)
- Spring 2005: *"3D Topographic data modelling: why rigidity is preferable to pragmatism"* (Cosit '05)

Proposed new approach

Full 3D TEN model

Two fundamental observations:

- ISO19101: a feature is an 'abstraction of real world phenomena'. These real world phenomena have by definition a volume
- Real world can be considered to be a volume partition (analogous to a planar partition: a set of non-overlapping volumes that form a closed modelled space)

Proposed new approach

Why a full 3D approach?

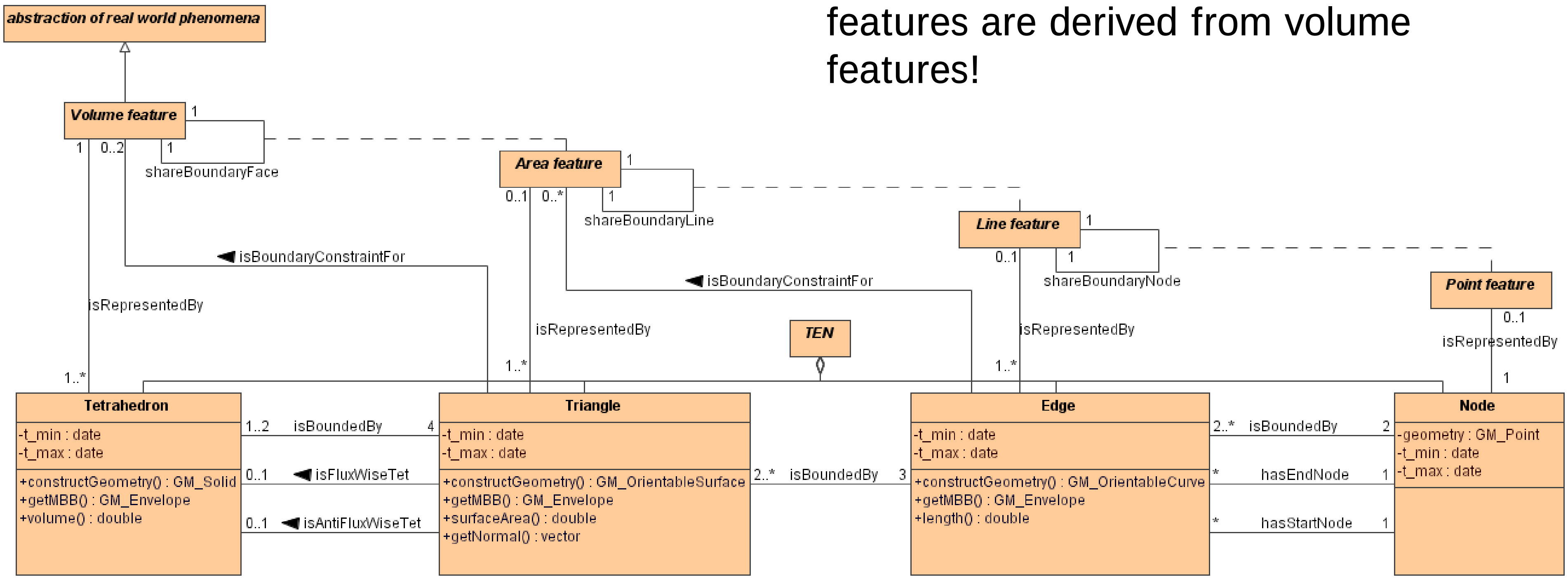
'Abstraction of real world phenomena' (ISO19101):

- until now:
  abstraction (simplification) → less dimensional representation
- when using meshes:
  simplification is in subdivision into easy-to-handle parts
  (analogously to Finite Element Method for solving Partial Differential Equations)

As a result: model only volume features in a volume partition (but handle polygon features as association class, derived from volume features, as faces mark transition between two volumes: for instance wall, earth surface)

Further research
UML model

Note: area, line and point features are derived from volume features!



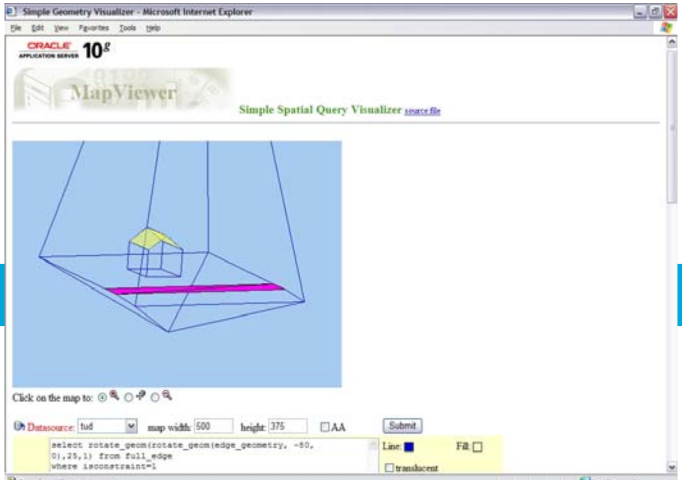
History in a nutshell:
Introduced to Poincaré

- July 2004: *“Realization of a three dimensional topographic terrain representation in an integrated TIN/TEN model”* (Agile 2005)
- September 2004: Linking TIN and TEN model (glue, stitch & nail)
- September 2005: *“3D Topographic data modelling: why rigidity is preferable to pragmatism”* (Cosit ’05)
- November 2005: one week visit to Oracle Spatial Development Centre

Oracle visit


Oracle visit

- Discussions on 3D
- First experiences ‘toy’ dataset: 56 tetrahedrons, 120 triangles, 83 edges, 20 nodes
- John Herring suggests Poincaré
- ‘Hands-on’ session
- Visualisation: Oracle MapViewer + function **`rotateGeom`**



History in a nutshell:
Poincaré-based approach

- July 2004: *"Realization of a three dimensional topographic terrain representation in an integrated TIN/TEN model"* (Agile 2005)
- September 2004: Linking TIN and TEN model (glue, stitch & nail)
- September 2005: *"3D Topographic data modelling: why rigidity is preferable to pragmatism"* (Cosit '05)
- November 2005: one week visit to Oracle Spatial Development Centre
- July 2006: *"A Tetrahedronized Irregular Network Based DBMS approach for 3D Topographic Data Modelling"* (SDH'06)
- September 2006: *"Updating Features in a TEN-based DBMS approach for 3D Topographic Data Modelling"* (GIScience'06)

TUDelft

Outline

- Introduction
- PhD project history in a nutshell
- Basics of modelling approach
- Poincaré's formalism of simplicial homology
- Concept of Poincaré-based TEN structure
- Implementation → illustration of concept
- Conclusions & future research
- Discussion

Basics of modelling approach:
Fundamental concept (1/2)

- Physical world objects have by definition a volumetric shape: there are no such things as point, line or area features! (*only point, line and area representations at a certain level of generalization*)

- The real world can be considered as a volume partition: a set of non-overlapping volumes that form a closed modeled space. As a consequence, objects like 'earth' or 'air' are explicitly part of the real world and thus have to be modeled.

Basics of modelling approach

Fundamental concept (2/2)

No area features at all?

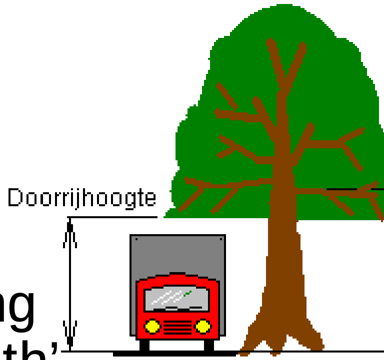
→ yes, but as derived features: area features (e.g. ‘wall’,’roof’) mark transition between volume features

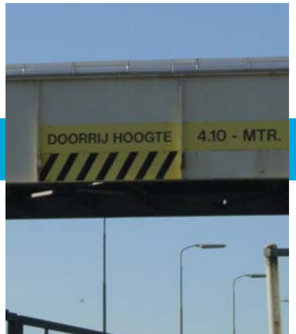
For instance:

- ‘Earth surface’ is transition between ‘earth’ and ‘air’
- ‘Wall/roof’ is transition between ‘house’ and ‘air’

But how to model for instance a road?

- Road is a volume (in which cars can drive without hitting something: ‘tunnel in the sky’). Transition between ‘earth’ and ‘road’ can be drawn with ‘road’ colors





Basics of modelling approach
TEN approach (1/2)

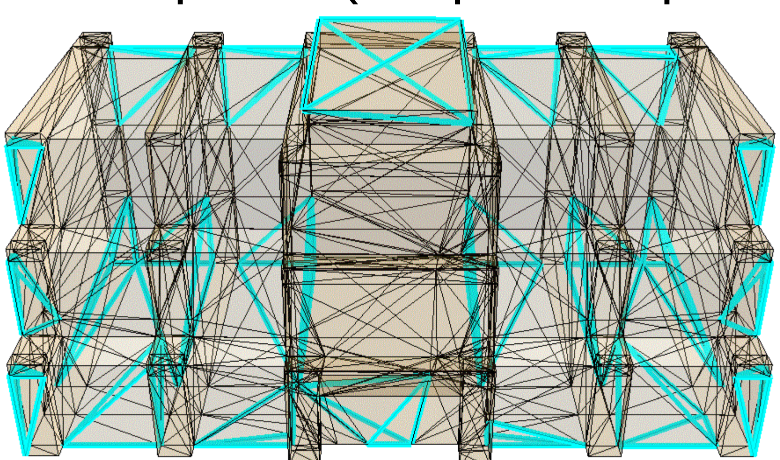
Important: data consistency and data analysis

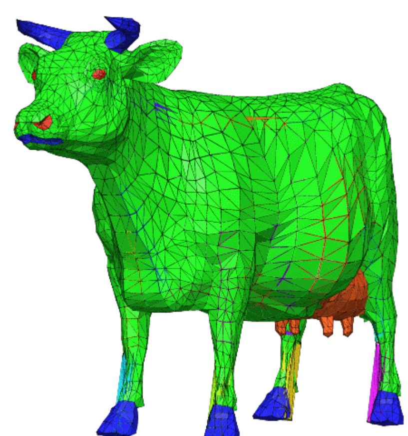
→ TEN structure well-suited:

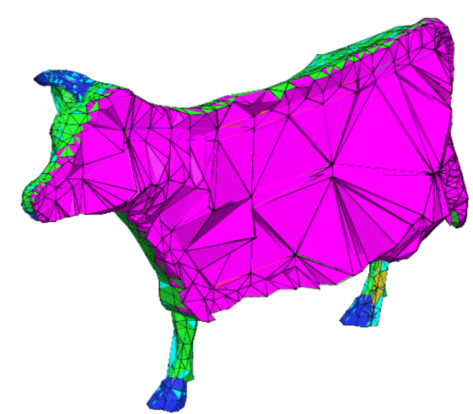
well-defined, only convex volumes, flat faces

In 3D (complex) shapes → subdivide in (many) tetrahedrons:

TEN is based on points, line segments, triangles and tetrahedrons:
simplexes (‘simplest shape in a given dimension’)





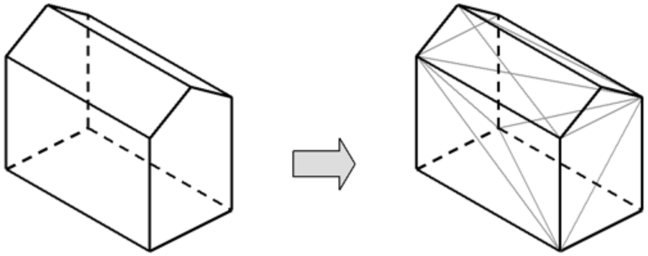


Basics of modelling approach
TEN approach (2/2)

Drawbacks:

- Complexity (but hide it from user...)
- Storage requirements:

Building as polyhedron	Building as TEN
(1 volume)	8 tetrahedrons
7 polygons	24 triangles
(15 edges)	25 edges
(10 points)	10 nodes



Outline

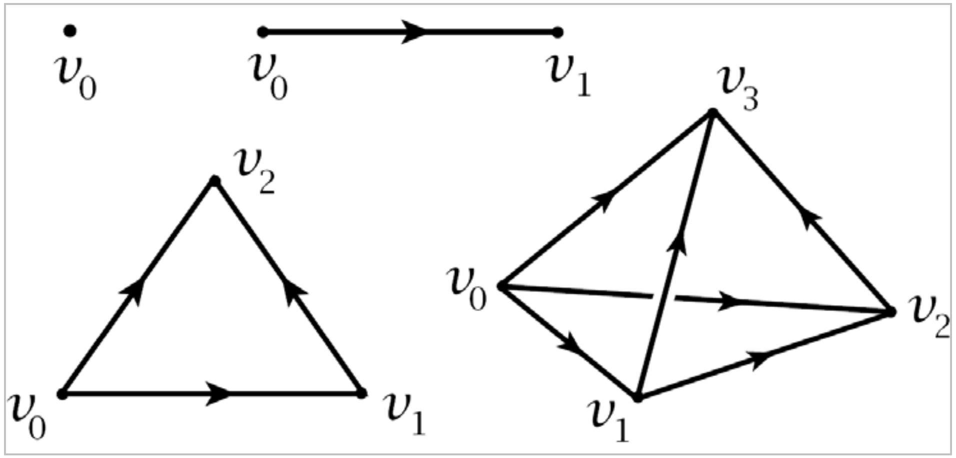
- Introduction
- PhD project history in a nutshell
- Basics of modelling approach
- Poincaré’s formalism of simplicial homology
- Concept of Poincaré-based TEN structure
- Implementation → illustration of concept
- Conclusions & future research
- Discussion

Poincaré simplicial homology (1/5)
Definition simplex

Jules Henri Poincaré
(1854-1912)

Solid mathematical foundation:

A n -simplex S_n is defined as smallest convex set in Euclidian space R^m of $n+1$ points $v_0, \ldots, v_n$ (which do not lie in a hyper plane of dimension less than n)



Poincaré simplicial homology (2/5)
Boundary operator

The boundary ∂ of simplex S_n is defined as sum of *(n-1)* dimensional simplexes (note that 'hat' means skip the node):

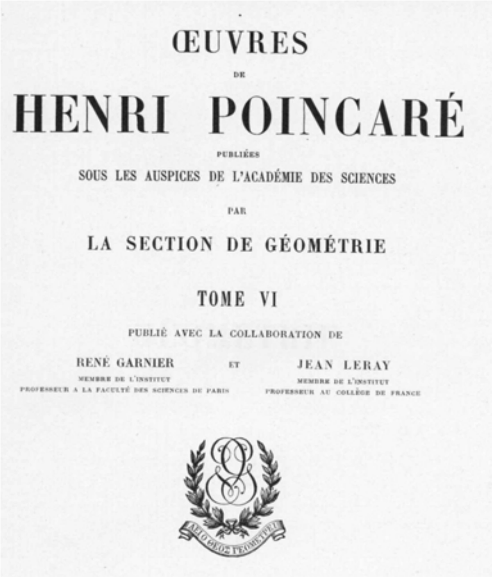
$$\partial S_n = \sum_{i=0}^{n} (-1)^i < v_0, \ldots, \hat{v}_i, \ldots, v_n >$$

remark: sum has n+1 terms

$$S_1 = <v_0, v_1> \qquad \partial S_1 = <v_1> - <v_0>$$

$$S_2 = <v_0, v_1, v_2> \qquad \partial S_2 = <v_1, v_2> - <v_0, v_2> + <v_0, v_1>$$

$$S_3 = <v_0, v_1, v_2, v_3> \qquad \partial S_3 = <v_1, v_2, v_3> - <v_0, v_2, v_3> + <v_0, v_1, v_3> - <v_0, v_1, v_2>$$


ŒUVRES

DE

HENRI POINCARÉ

PUBLIÉES

SOUS LES AUSPICES DE L'ACADÉMIE DES SCIENCES

PAR

LA SECTION DE GÉOMÉTRIE

TOME VI

PUBLIÉ AVEC LA COLLABORATION DE

RENÉ GARNIER ET JEAN LERAY

MEMBRE DE L'INSTITUT, PROFESSEUR A LA FACULTÉ DES SCIENCES DE PARIS

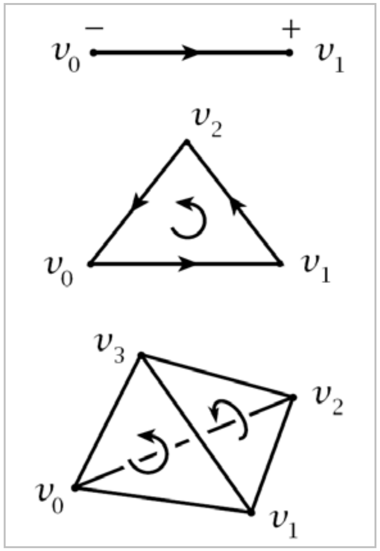
MEMBRE DE L'INSTITUT, PROFESSEUR AU COLLÈGE DE FRANCE

Poincaré simplicial homology (3 / 5)
Simplex construction

$$S_n \text{ has } \binom{n+1}{p+1} \text{ faces of dimension } p \text{ with } (0 \le p < n)$$

- 2D: this means that triangle (S_2) has 3 edges (S_1) and 3 nodes (S_0)
- 3D: this means that tetrahedron (S_3) has 4 triangles (S_2), 6 edges (S_1) and 4 nodes (S_0)

Poincaré simplicial homology (4/5)
Orientation of boundaries

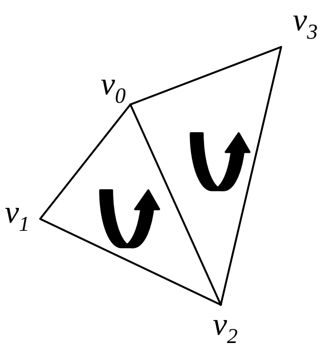


With $(n+1)$ points, there are $(n+1)!$ permutations of these points. In 3D for the 4 simplexes this means 1, 2, 6 and 24 options (S_0 obvious):

- For S_1 the two permutations are $\langle v_0, v_1 \rangle$ and $\langle v_1, v_0 \rangle$ (one positive and one negative $\langle v_0, v_1 \rangle = -\langle v_1, v_0 \rangle$)
- For S_2 there are 6: $\langle v_0,v_1,v_2 \rangle$, $\langle v_1,v_2,v_0 \rangle$, $\langle v_2,v_0,v_1 \rangle$, $\langle v_2,v_1,v_0 \rangle$, $\langle v_0,v_2,v_1 \rangle$, and $\langle v_1,v_0,v_2 \rangle$. First 3 opposite orientation from last 3, e.g. $\langle v_0,v_1,v_2 \rangle = -\langle v_2,v_1,v_0 \rangle$. counter clockwise (+) and the negative orientation is clockwise (-)
- For S_3 there are 24, of which 12 with all normal vectors outside (+) and 12 others with all normal vectors inside (-)!

Poincaré simplicial homology (5/5)

Simplicial complexes

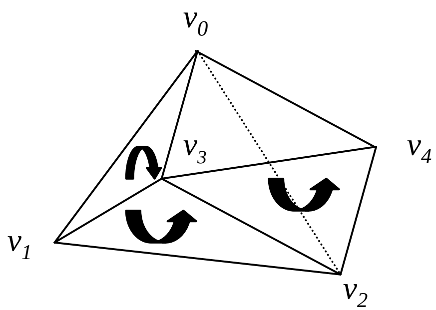


$$S_{21} = < v_0, v_1, v_2 > \text{and } S_{22} = < v_0, v_2, v_3 >$$

$$C_2 = < v_1, v_2 > - < v_0, v_2 > + < v_0, v_1 >$$

$$+ < v_2, v_3 > - < v_0, v_3 > + < v_0, v_2 >$$

$$= < v_1, v_2 > + < v_0, v_1 > + < v_2, v_3 > + < v_3, v_0 >$$



$$S_{31} = < v_0, v_1, v_2, v_3 > \text{and } S_{32} = < v_0, v_2, v_4, v_3 >$$

$$C_3 = < v_1, v_2, v_3 > - < v_0, v_2, v_3 > + < v_0, v_1, v_3 >$$

$$- < v_0, v_1, v_2 > + < v_2, v_4, v_3 > - < v_0, v_4, v_3 >$$

$$+ < v_0, v_2, v_3 > - < v_0, v_2, v_4 >$$

$$= < v_1, v_2, v_3 > + < v_0, v_1, v_3 > - < v_0, v_1, v_2 >$$

$$+ < v_2, v_4, v_3 > - < v_0, v_4, v_3 > - < v_0, v_2, v_4 >$$

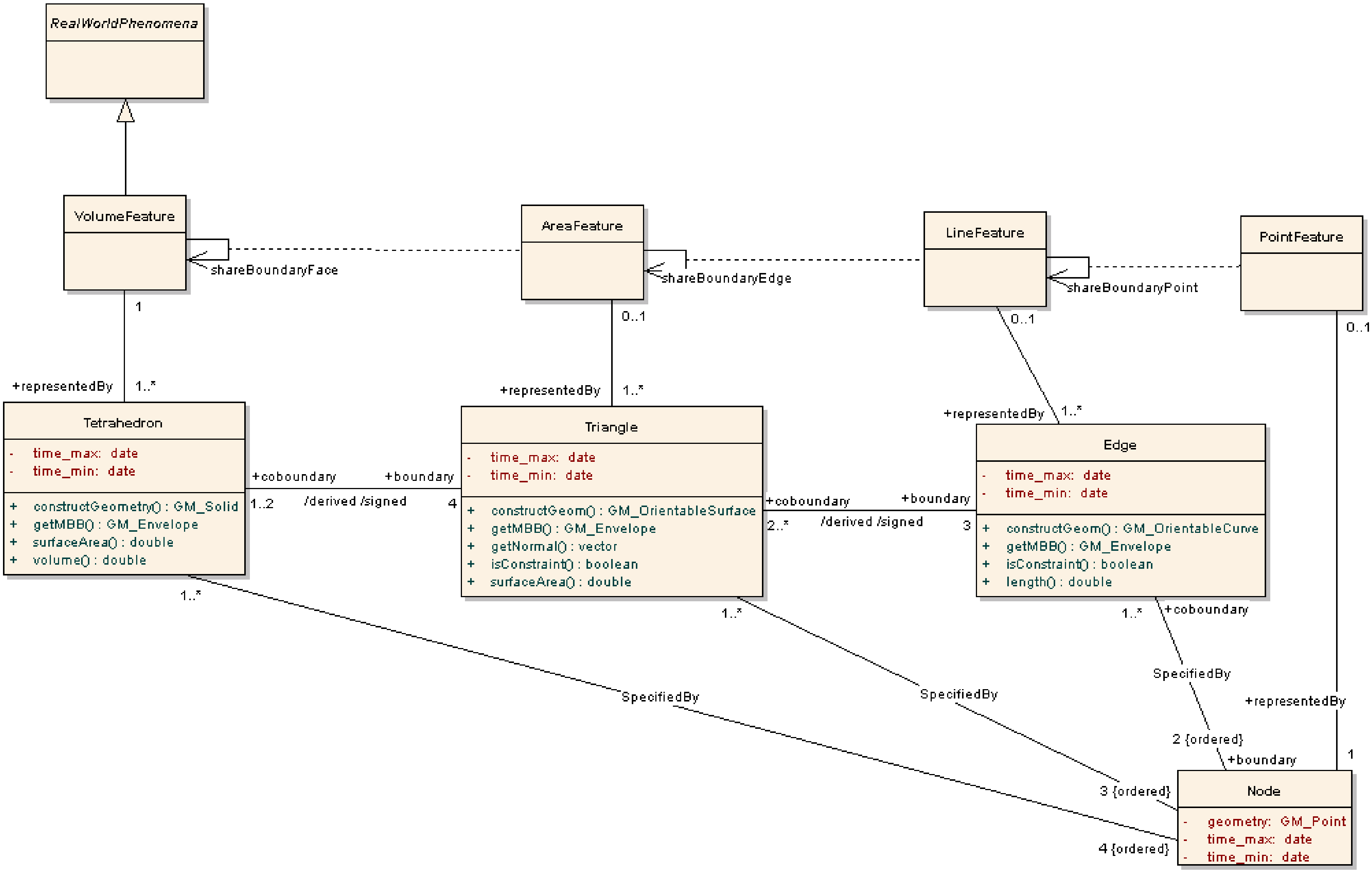
Outline

- Introduction
- PhD project history in a nutshell
- Basics of modelling approach
- Poincaré's formalism of simplicial homology
- Concept of Poincaré-based TEN structure
- Implementation → illustration of concept
- Conclusions & future research
- Discussion

TEN approach using simplicial homology
First ideas

- Describe all simplexes by their vertices
- Store tetrahedrons and vertices in table
- Derive triangles and edges (views) using the boundary operator
- Use signed and oriented simplexes

In UML this looks like...



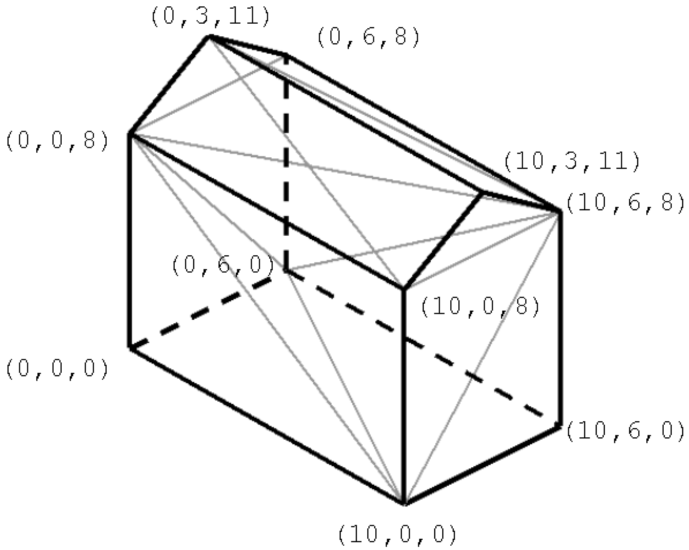
RealWorldPhenomena
VolumeFeature
AreaFeature
LineFeature
PointFeature
shareBoundaryFace
shareBoundaryEdge
shareBoundaryPoint
1
0..1
0..1
0..1
+representedBy
1..*
+representedBy
1..*
+representedBy
1..*
Tetrahedron
- time_max: date
- time_min: date
+ constructGeometry() : GM_Solid
+ getMBB() : GM_Envelope
+ surfaceArea() : double
+ volume() : double
+coboundary
+boundary
1..2
/derived /signed
4
Triangle
- time_max: date
- time_min: date
+ constructGeom() : GM_OrientableSurface
+ getMBB() : GM_Envelope
+ getNormal() : vector
+ isConstraint() : boolean
+ surfaceArea() : double
+coboundary
+boundary
2..*
/derived /signed
3
Edge
- time_max: date
- time_min: date
+ constructGeom() : GM_OrientableCurve
+ getMBB() : GM_Envelope
+ isConstraint() : boolean
+ length() : double
1..*
1..*
1..*
+coboundary
SpecifiedBy
SpecifiedBy
SpecifiedBy
+representedBy
2 {ordered}
+boundary
1
3 {ordered}
4 {ordered}
Node
- geometry: GM_Point
- time_max: date
- time_min: date

TEN approach using simplicial homology

Current ideas

- Encode vertex coordinates
- Describe all simplexes by their coded vertices
- Store tetrahedrons in table
- Derive triangles, edges and nodes (views) using the boundary operator
- Use signed and oriented simplexes

```
(00,00,00)(00,00,08)(00,06,00)(10,00,00)
(00,00,08)(00,03,11)(00,06,08)(10,06,08)
(00,00,08)(00,03,11)(10,00,08)(10,06,08)
(00,00,08)(00,06,00)(00,06,08)(10,06,08)
(00,00,08)(00,06,00)(10,00,00)(10,06,08)
(00,00,08)(10,00,00)(10,00,08)(10,06,08)
(00,03,11)(10,00,08)(10,03,11)(10,06,08)
(00,06,00)(10,00,00)(10,06,00)(10,06,08)
```



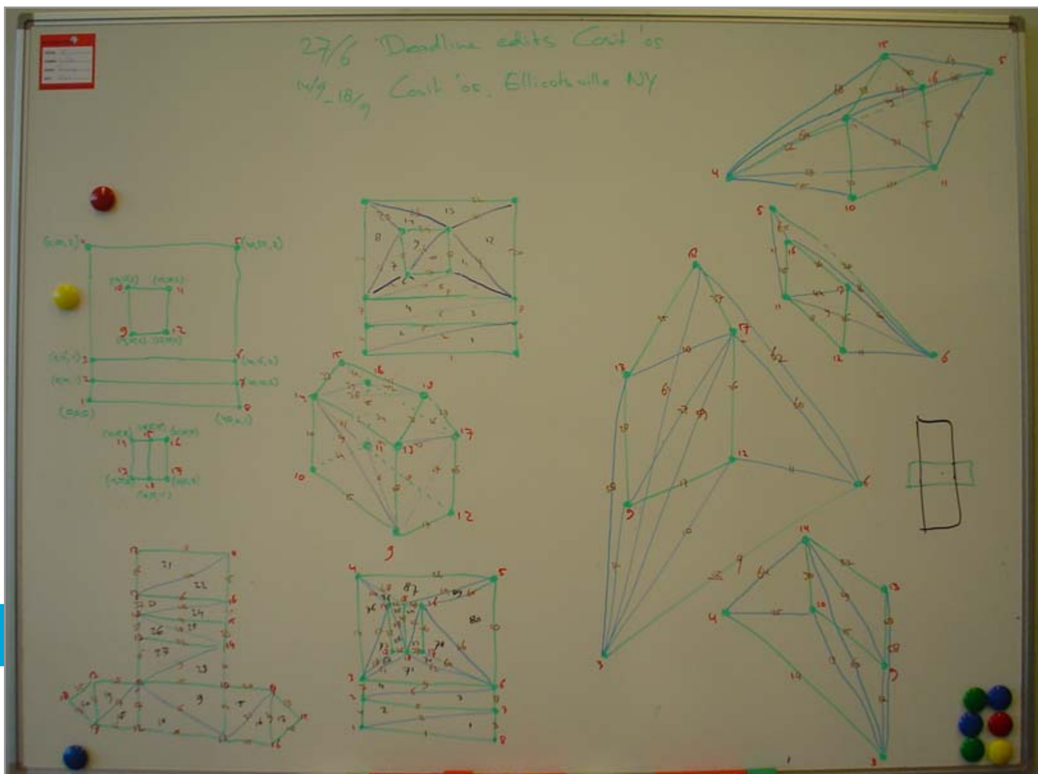
Outline

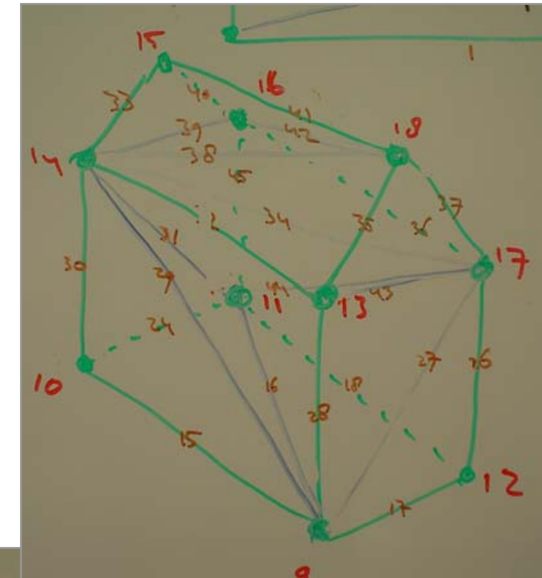
- Introduction
- PhD project history in a nutshell
- Basics of modelling approach
- Poincaré’s formalism of simplicial homology
- Concept of Poincaré-based TEN structure
- **Implementation → illustration of concept**
- Conclusions & future research
- Discussion

Implementation
Tetrahedronization

At this moment: tetrahedronization by hand

'Toy' dataset: 56 tetrahedrons, 120 triangles, 83 edges, 20 nodes





Implementation

Loading data in tetrahedron table

```
CREATE TABLE tetrahedron( tetcode NVARCHAR2(100));
```

```
LOAD DATA INFILE 'data/miniset.data' APPEND
INTO TABLE tetrahedron
fields terminated by ' '
( tetcode )
```

Result: table with tetrahedrons

```
014025012014035012022035012014035018003
014025012022035012022025012022025018003
014025012014025018014035018022025018003
022035012014035018022035018022025018003
014025012022035012014035018022025018003
014025018014035018022025018018025021003
014035018018035021022035018018025021003
014035018022035018022025018018025021003
020055000000000010040010012040000011001
020055000000000010000010011040010012001
020055000000010011040016012040010012004
020055000000010011000016011040016012004
020055000000016011040016012022025012001
020055000000016011014025012022025012001
020055000000016011014025012014035012001
020055000000016011000050012014035012001
```

Implementation

Fixing orientation tetrahedrons

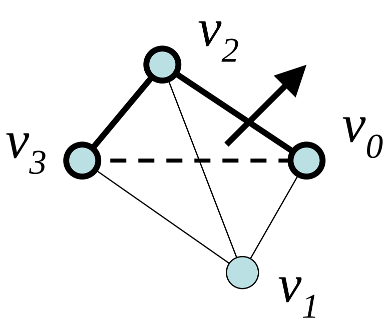
Objective: all tetrahedrons oriented outwards

→ each boundary triangle appears two times: 1x pos., 1x neg.

```
CREATE OR REPLACE PROCEDURE tettableoutwards
(…)
checkorientation(codelength,currenttetcode,bool);
IF (bool = 0) THEN
   permutation12(codelength,currenttetcode,newtetcode);
   UPDATE tetrahedron SET tetcode=newtetcode WHERE CURRENT OF tetcur;
(…)
END;
```

`checkorientation` : angle normal vector and vector to opposite point

`permutation12` : $(v_0,v_1,v_2,v_3) \rightarrow (v_1,v_0,v_2,v_3)$



Implementation

Deriving boundary triangles

Applying boundary operator: $\partial S_n = \sum_{i=0}^{n} (-1)^i < v_0, \ldots, \hat{v}_i, \ldots, v_n >$

Procedure:

```
CREATE OR REPLACE PROCEDURE deriveboundarytriangles(
    (…)
   a := (SUBSTR(tetcode,1,3*codelength));
   b := (SUBSTR(tetcode,1+3*codelength,3*codelength));
   c := (SUBSTR(tetcode,1+6*codelength,3*codelength));
   d := (SUBSTR(tetcode,1+9*codelength,3*codelength));
   id := (SUBSTR(tetcode,1+12*codelength));
   ordertriangle(codelength,'+'||b||c||d||id, tricode1);
   ordertriangle(codelength,'-'||a||c||d||id, tricode2);
   ordertriangle(codelength,'+'||a||b||d||id, tricode3);
   ordertriangle(codelength,'-'||a||b||c||id, tricode4);
     (…)
```

Example:

```
014025012014035012022035012014035018003

a : 014025012
b : 014035012
c : 022035012
d : 014035018
id: 003
```

$\left.\right\}$ see next slide for results of **`ordertriangle`**

Note: triangles inherit ID of object which is represented by the tetrahedron of which they are a boundary!

Implementation
Ordering triangles

Objective: gain control over which permutation is used

`ordertriangle` rewrites in form <a,b,c> such that a<b<c

```
+01403501202203501201403501800З
+ 014035012 022035012 014035018 003
- 014035012 014035018 022035012 003
-014035012014035018022035012003
```

$\langle v_0,v_1,v_2\rangle$	-
$\langle v_0,v_2,v_1\rangle$	+
$\langle v_2,v_0,v_1\rangle$	-
$\langle v_2,v_1,v_0\rangle$	+
$\langle v_1,v_2,v_0\rangle$	-
$\langle v_1,v_0,v_2\rangle$	+

Implementation

Creating view triangle

```
CREATE OR REPLACE VIEW triangle AS
   SELECT deriveboundarytriangle1(3,tetcode) tricode FROM tetrahedron
   UNION ALL
   SELECT deriveboundarytriangle2(3,tetcode) tricode FROM tetrahedron
   UNION ALL
   SELECT deriveboundarytriangle3(3,tetcode) tricode FROM tetrahedron
   UNION ALL
   SELECT deriveboundarytriangle4(3,tetcode) tricode FROM tetrahedron;
```

Four functions: first gives first boundary, etc.

Result: #triangles = 4 * #tetrahedrons

Every triangle appears two times: once with sign +, once with sign –
(and NOT in a permutated form → due to **`ordertriangle`**!)

Implementation

Creating view constrainedtriangle

```
CREATE OR REPLACE VIEW constrainedtriangle AS
   SELECT t1.tricode tricode FROM triangle t1
   WHERE NOT EXISTS (SELECT t2.tricode FROM triangle t2 WHERE t1.tricode =
    t2.tricode*-1);
```

Well, not every triangle appears two times:

A constrained triangle is a boundary between two objects

→ two different id’s → two different triangle codes!

Example: in -1,7,2,-7,-3,1 the constrained triangles are 2 and -3

Implementation

Creating views edge, constrainededge

In current implementation edges are undirected en do not inherit object id's (as no application for this is identified at the moment)

```
CREATE OR REPLACE VIEW edge AS
   SELECT DISTINCT deriveabsboundaryedge1(3,tricode) edcode FROM triangle
   UNION
   SELECT DISTINCT deriveabsboundaryedge2(3,tricode) edcode FROM triangle
   UNION
   SELECT DISTINCT deriveabsboundaryedge3(3,tricode) edcode FROM triangle;
```

All boundary edges from constrained triangles are constrained edges:

```
CREATE OR REPLACE VIEW constrainedtriangle AS
   SELECT t1.tricode tricode FROM triangle t1
   WHERE NOT EXISTS (SELECT t2.tricode FROM triangle t2 WHERE t1.tricode =
t2.tricode*-1);
```

Implementation
Validating the structure (1/3)

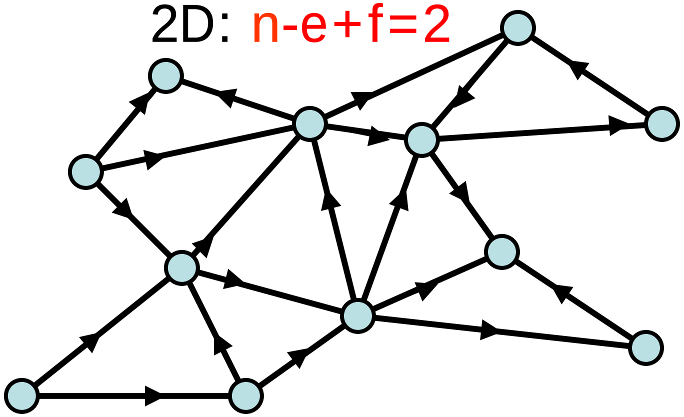
Leonard Euler, 1707-1783

After creating node view the structure can be validated:

3D Euler-Poincaré

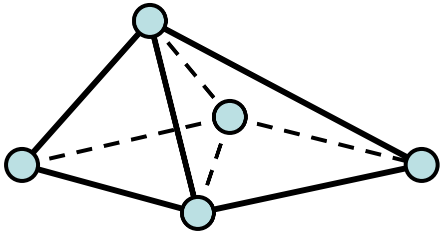
$$N - E + F - V = 0$$

2D: n-e+f=2



(12-21+11=2)

3D: n-e+f-v=0



(5-9+7-3=0)

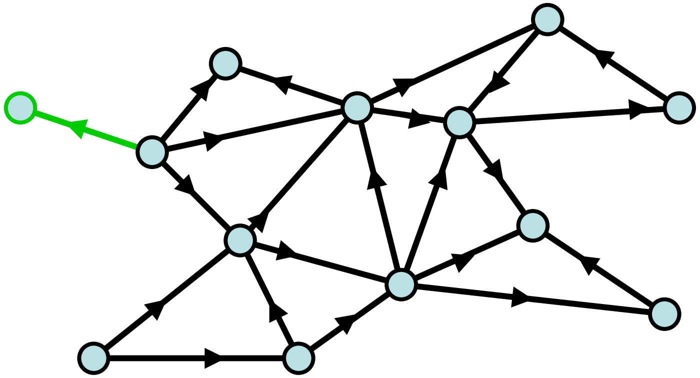
Implementation

Validating the structure (2/3)

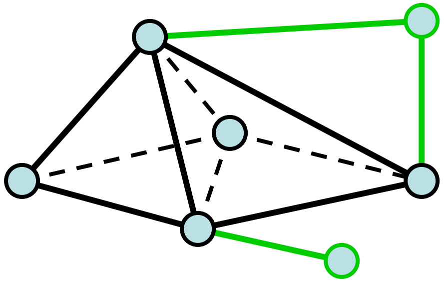
Leonard Euler, 1707-1783

Limitations:

3D Euler-Poincaré holds for all simplicial complexes, including complexes build up of simplexes of different dimension (i.e. dangling edges and faces are allowed)



(13-22+11=2)



(7-12+8-3=0)

Implementation

Validating the structure (3/3)

3D Euler-Poincaré for current dataset:

$$N - E + F - V = 0$$

20 –83 + 224 –57 <> 0 !

→ View triangle contains duals, need to be excluded from count:

```
SELECT COUNT(DISTINCT ABS(removeobjectid(3,tricode)))
   INTO numtri FROM triangle;
```

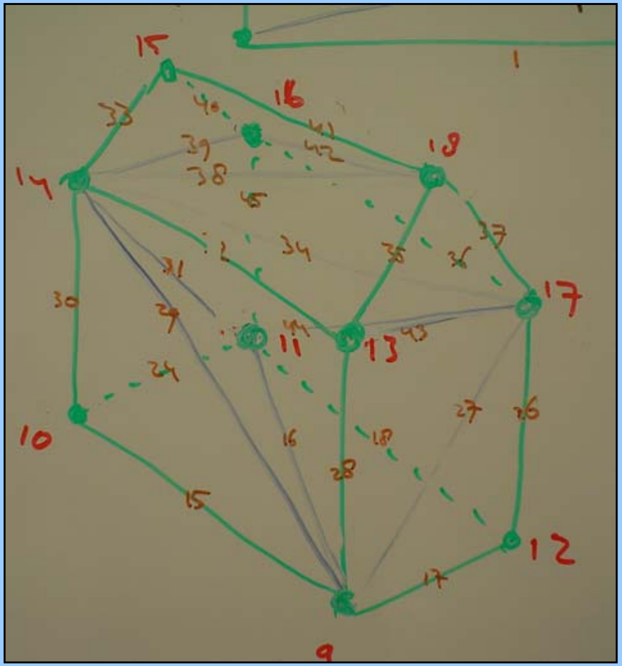
```
> select count(*) from tetrahedron;
  COUNT(*)
----------
        56
> select count(*) from triangle;
  COUNT(*)
----------
       224
> select count(*) from edge;
  COUNT(*)
----------
        83
> select count(*) from node;
  COUNT(*)
----------
        20
```

Implementation

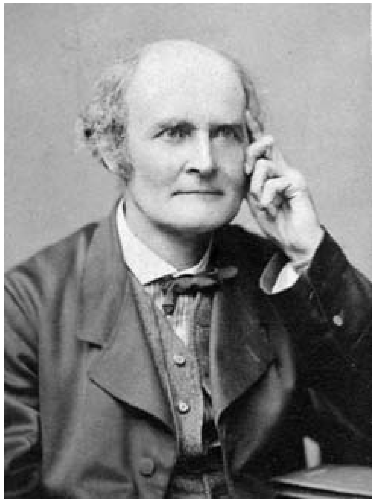
Query and analysis (1/2)

Query: boundary triangulation of building (object ID = 3)

```
SQL> select tricode from constrainedtriangle where getobjectid(3,tricode)=3;
TRICODE
-----------------------------------------------------------------------------------
-014025012014025018022025018003
-014025018018025021022025018003
-018025021018035021022035018003
+014025012022025012022025018003
-014025012014035012014035018003
+018025021022025018022035018003
-014025012022025012022035012003
+014025012014025018014035018003
+014025012014035012022035012003
-014035018022035012022035018003
+014035012014035018022035012003
+014035018018025021018035021003
+022025018022035012022035018003
+014035018018035021022035018003
-022025012022025018022035012003
-014025018014035018018025021003
16 rows selected.
Elapsed: 00:00:00.09
```



Implementation
Query and analysis (2/2)


Arthur Cayley, 1821-1895

Volume of house, surface of boundary of house:

`simplexvolume()` implements Cayley-Menger determinant

Cayley-Menger determinant gives the volume of a simplex in *j* dimensions.

j=2:
$$-16\,\Delta^2 = \begin{vmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & c^2 & b^2 \\ 1 & c^2 & 0 & a^2 \\ 1 & b^2 & a^2 & 0 \end{vmatrix}.$$

j=3:
$$288\,V^2 = \begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & d_{12}^2 & d_{13}^2 & d_{14}^2 \\ 1 & d_{21}^2 & 0 & d_{23}^2 & d_{24}^2 \\ 1 & d_{31}^2 & d_{32}^2 & 0 & d_{34}^2 \\ 1 & d_{41}^2 & d_{42}^2 & d_{43}^2 & 0 \end{vmatrix}.$$

(with *a*,*b*,c and d_{ij} length of simplex edges)

Implementation
Performance

Indexing:

- Primary index: sort coded simplexes
- Secondary index: R-tree on tetrahedrons, using **`gettetrahedronmbb(), gettrianglembb()`** etc.

Coding:

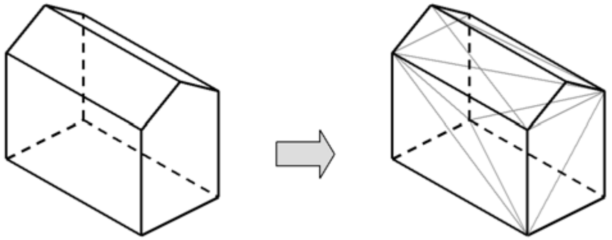
- More work on encoding coordinates: bitwise interleaving, ...

Implementation
Final thougth

Polyhedron vs. TEN:

does a TEN really require that much more storage space than polyhedrons?

Building as polyhedron	Building as TEN
(1 volume)	8 tetrahedrons
7 faces	(24 triangles)
(15 edges)	(25 edges)
(10 points)	(10 nodes)



Conclusions & future research

Result:

- Topological 3D (TEN) data structure, stored in one single-column table (!)
- with advantages of TEN, but not its drawbacks (?)
- based on a solid theoretical foundation (100 years old math)

Future research ideas:

- Inclusion of incremental constrained Delaunay tetrahedronization
- Test with real data (Den Bosch case?)
- Compare to Calin/Oracle11 polyhedron approach
- History (?)

TU Delft

Discussion
(with the living...)