3D and exact geometries for PostGIS

FOSDEM PGDay

02-01-2013 – Hugo Mercier / Oslandia
Oslandia

PostGIS, QGIS, Mapserver suite
Training
Support
Development
FEDER-funded program e-PLU
City modelling applications
3D spatial operations
  PostGIS geometries can carry a z coordinate
What about spatial processing?
IGN – Oslandia collaboration
GEOS (PostGIS geometry backend) is 2D only
Appealing candidate : CGAL

Modern C++ framework
Lots of 2D/3D algorithms already implemented
Exact computational model !
Does it perform well ?
Exact computation

CGAL is templated by geometric 'Kernel's
Must use an 'exact' kernel for constructions
  Arbitrary precision numbers
  Lazy evaluation of an expression tree
  Interval arithmetics

<example?>
Design of an OGC Simple Features compliant framework on top of CGAL
Our own PostGIS branch (postgis-sfcdnal)
Currently supported:

- 2D and 3D intersection (including solids)
- 2D and 3D intersection test (including solids)
- 2D and 3D convex hull
- 2D and 3D triangulations
- 3D extrusion
- 2D and 3D distances
- (in progress) buffers
Postgis-sfcsgal:

Optional support for SFCGAL functions
Using the 'sfcsagal' schema

E.g.:

```
SELECT sfcsagal.ST_Intersects( g1, g2 )
SELECT sfcsagal.ST_3DIntersection(g1, g2)
...
```
Performance comparison

PostGIS based
2D Only
Varying geometry's number of points
SFCGAL vs GEOS

The graph shows the comparison between SFCGAL and GEOS in terms of time (in seconds) required to perform intersection operations as a function of the number of points. The graph indicates that SFCGAL performs more efficiently than GEOS, especially as the number of points increases.
SFCGAL vs GEOS

The graph shows a comparison between SFCGAL and GEOS for the function `intersects_polygon_polygon`.

The x-axis represents the number of points, ranging from 0 to 200. The y-axis represents the time in seconds, ranging from 0.1 to 0.7.

Two lines are plotted:
- The blue line represents GEOS.
- The green line represents SFCGAL.

As the number of points increases, the time taken by both GEOS and SFCGAL also increases, with SFCGAL generally taking more time than GEOS for the same number of points.
SFCGAL vs GEOS

The graph compares the area_polygon calculation time for GEOS and SFCGAL as a function of the number of points. GEOS shows a nearly constant time with some variation, while SFCGAL exhibits an increasing trend with the number of points.
Results are very promising

SFCGAL is sometimes better, sometimes worse

Considering SFCGAL is way less mature than GEOS

Comparable behaviour and space for improvements!
Couple PostGIS 3D to a 3D viewer

SFCGAL viewer (https://vimeo.com/58523983)
3D view

QGIS with the Globe plugin (https://vimeo.com/54776907)
Precision issues

Do they intersect?
Precision issues

Do they intersect?

```sql
SELECT
   ST_Intersection(
      'LINESTRING(0 0,2 1)'::geometry,
      'LINESTRING(1 0,0 1)'::geometry,
      'LINESTRING(0 0,2 1)'::geometry)

st_intersects
--------------
  f
(1 row)
```

Should be true!

GEOS only supports 'double' numbers
Support for exact geometries

New 'exact_geometry' type

Coordinates stored with arbitrary precision

Serialization/deserialization process

```sql
SELECT Sfcgal.ST_Intersection('LINESTRING(0 0,2 1)':'exact_geometry', 'LINESTRING(1 0,0 1)':'exact_geometry')
```

```
st_intersection
-------------------
POINT(2/3 1/3)
```

```sql
SELECT Sfcgal.ST_Intersects(Sfcgal.ST_Intersection('LINESTRING(0 0,2 1)':'exact_geometry', 'LINESTRING(1 0,0 1)':'exact_geometry'), 'LINESTRING(0 0,2 1)':'exact_geometry');
```

```
st_intersects
-------------
t
(1 row)
```
'exact_geometry' serialization is slow!
Comparing 4 chained 'noop' functions
SELECT ST_Copy(ST_Copy(ST_Copy(ST_Copy( g ))))
Serialization performances

![Graph showing serialization performances](image)
Btw, do we need to serialize?
Not if you only need temporary results
\[ \text{ST}_f1(\text{ST}_f2(\text{ST}_f3(g : \text{geometry}))) \]

New type 'ref\_geometry'
Complex C++ objects (SFCGAL::Geometry*) can be created and passed by reference
Referenced geometries

Native vs. referenced serialization, 1000 geoms

- Native inexact
- Referenced exact

Time (s)

# of points
When to deallocate?

Everything allocated by palloc() will be freed on Memory Context's reset/deletion

C++ objects need **destruction, not only deallocation**!

Solution

Use a child context with your own deletion method
Where to allocate?

The parent context where to allocate is important.
If it vanishes too fast, we loose our objects.
If it lives too long, we explode memory.

Current rule of thumb:
Attach to the ExprContext when we can.
Attach to a long-living context otherwise (MessageContext).
Referenced geometries
Cannot be stored!

```sql
SELECT sfcgal.ST_Intersects(
    Sfcgal.ST_Intersection(
        'LINESTRING(0 0,2 1) '::ref_geometry,
        'LINESTRING(1 0,0 1) '::ref_geometry),
        'LINESTRING(0 0,2 1) '::ref_geometry);
```

```
CREATE TEMPORARY TABLE t AS
    SELECT 'POINT(0 0) '::ref_geometry;

SELECT * FROM t;
NOTICE: Referenced geometries must not be stored
    ref_geometry
--------------
    -deleted-
(1 row)
```
Control over serialization / deserialization
  
  Through conversion functions

  \[
  \text{ST\_Geometry}( \text{ref\_geometry} ) : \text{geometry} \\
  \text{ST\_RefGeometry}( \text{geometry} ) : \text{ref\_geometry}
  \]

\[
\Rightarrow
\]

\[
\text{SELECT } \text{ST\_Geometry}( \\
  \text{ST\_f1}( \\
  \text{ST\_f2}( \\
  \text{ST\_f3}( \text{ST\_RefGeometry}( g ) \\
  )))))
\]
Conclusion

High potential!
3D spatial processing
Exact computation
With good performances
Work in progress

- PostGIS integration
- Referenced geometry testing
- Cache mechanism
- Spatial operations (boolean set)
- QGIS integration
http://www.oslandia.com

On github

Oslandia/SFCGAL
Oslandia/postgis-sfcgal

hugo.mercier@oslandia.com