Fixstars Geometric Performance Primitives

World's Fastest Multi-Core Geometry Engine

Version 1.0

1 Introduction

Computational geometry is a branch of computer science devoted to the study of algorithms that can be described in terms of geometry. Some purely geometrical problems arise out of the study of these algorithms and such problems are also considered to be part of computational geometry.

The main impetus for the development of computational geometry as a discipline was progress in computer graphics and computer-aided design and manufacturing (CAD/CAM). However, many problems in computational geometry are classical in nature, and come from mathematical visualization.

Other important applications of computational geometry include robotics (motion planning and visibility problems), geographic information systems (GIS) (geometrical location, search, and route planning), integrated circuit design (IC geometry design and verification), computer-aided engineering (CAE) (mesh generation), computer vision (3D reconstruction) [see 4-Resources below].

Fixstars has developed a computational geometry engine for GPUs (CUDA Thrust) and multi-core CPUs (OpenMP) designated “GPP (Geometric Performance Primitives).” GPP is the only computational geometry engine that's optimized for truly massive multi-core parallelization. This gives GPP incredible speed and the ability to process truly enormous datasets.

2 Specification

2.1 Functions

The current version of GPP has the following functions:

1. line segment intersection

2. snap rounding for robustness

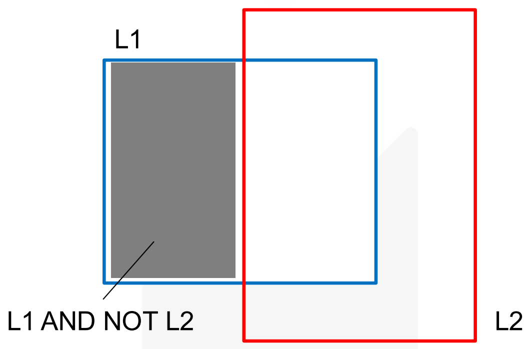
3. map overlay + filtering for boolean operations

The filtering function takes a predicate function as argument, that essentially supports anything that can be done in C/C++. For example, a predicate function for an "L1 AND NOT L2" boolean operation in multiple-layer mode may look like this:

```
bool predicate(Label mask, void *userData) {
    return (mask & 0x1) && !(mask & 0x2);
}
```

To output areas common to more than one polygon in single-layer mode, the predicate would look like this:

```
bool predicate(Label count, void *userData) {
    return count > 1;
}
```



GPP supports multiple layers, but only 8 are currently practical. (This may be extended in the future.) All polygons are free to overlap and self-overlap. GPP will take care of merging together all output faces that touch each other.

4. Polygon Relation Check

The polygon relation check operation ("RELATE") can receive two polygon layers as arguments, the "candidate" layer and the "filter" layer, or only one layer. The former operation result is a subset of candidate polygons, selected based on their interaction with filter polygons, while the latter outputs all relations. GPP actually compares polygon edges and for each pair of interacting edges returns a value describing the relation with the following bit flags:

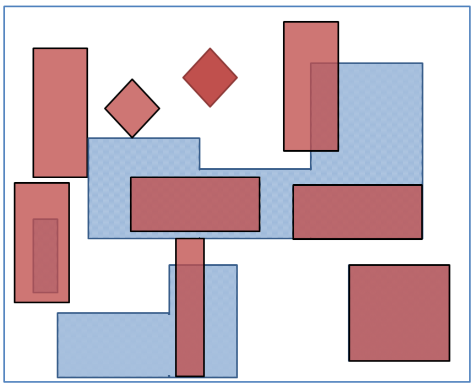
```
    ZONE_II = 0x01, ///< interior of edge1 intersects interior of edge2

    ZONE_IB = 0x02, ///< interior of edge1 intersects boundary of edge2

    ZONE_BI = 0x04, ///< boundary of edge1 intersects interior of edge2

    ZONE_BB = 0x08, ///< boundary of edge1 intersects boundary of edge2

    ZONE_O1 = 0x10, ///< order of edges is: edge1 left of edge2

    ZONE_O2 = 0x20, ///< order of edges is: edge2 left of edge1

    ZONE_D1 = 0x40, ///< edge1 direction is up (or right if horizontal)

    ZONE_D2 = 0x80, ///< edge2 direction is up (or right if horizontal)
```

As with boolean operations, the user can further provide a predicate function, but this time returning a three-valued boolean, allowing to classify polygons based solely on their edges relations, for performance reasons. As example, to obtain all intersecting polygons, such a "TriboolPredicate" would look like this:

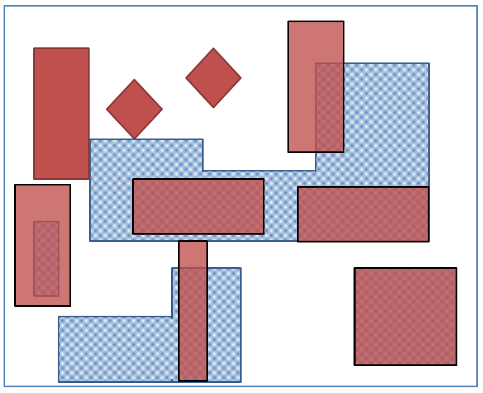
```
Tribool intersects(Label relation, void*  userData) {

    bool intersect = relation & (ZONE_II | ZONE_IB | ZONE_BI | ZONE_BB);

    bool contained1 = (relation & ZONE_O1) && !(relation & ZONE_D2);

    bool contained2 = (relation & ZONE_O2) && !(relation & ZONE_D1);

    if (intersect || contained1 || contained2) {

        return TRIBOOL_TRUE;

    } else {

        return TRIBOOL_MAYBE;

    }

}
```

With this and a few other flags to invert the role of the layers or to take the complement of resulting edges, it is possible to implement any kind of relations. However, GPP also comes built-in with the following commonly used spatial queries, whose result is a subset of candidate polygons, selected based on their interaction with filter polygons according to one of the following criteria (candidate layer is red, filter layer is blue, and result is outlined):

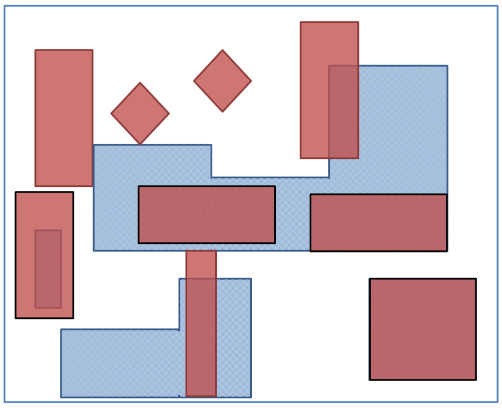
Intersect: Candidate polygon intersects (has at least one common point with) any filter polygon



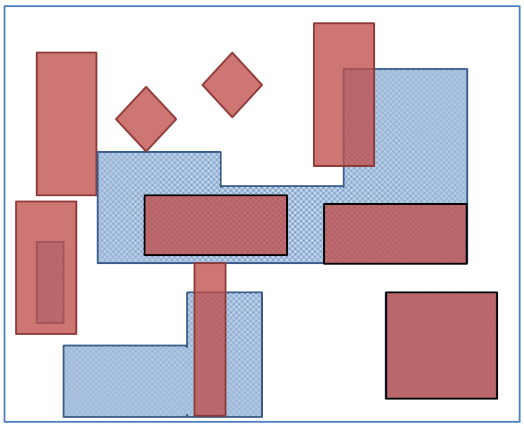
Overlap: Candidate polygon overlaps (has non-zero intersection area with) any filter polygon



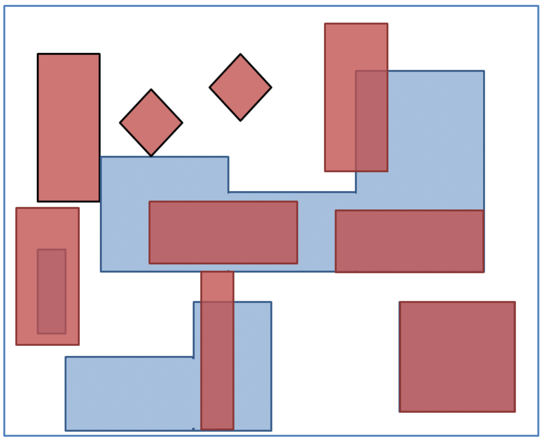
Enclose: Candidate polygon encloses any filter polygon



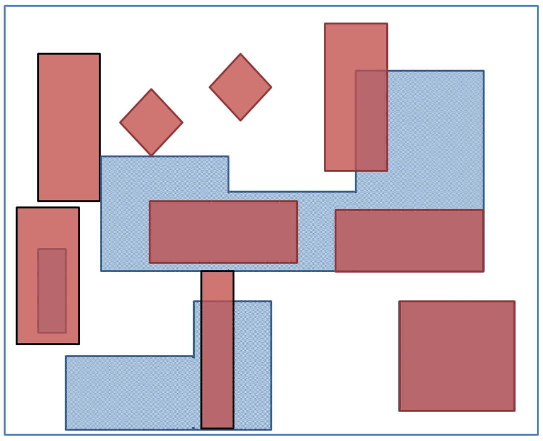
Inside: Candidate polygon is inside any filter polygon



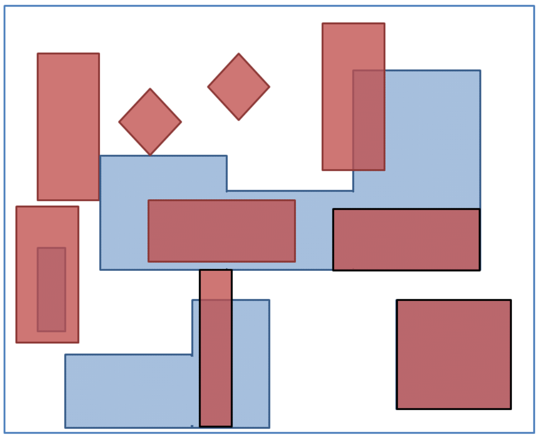
Outside: Candidate polygon is outside all filter polygons



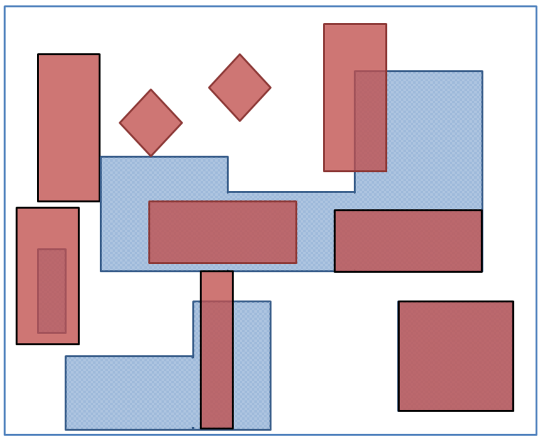
Abut Outside: Candidate polygon abuts any filter polygon from outside



Abut Inside: Candidate polygon abuts any filter polygon from inside



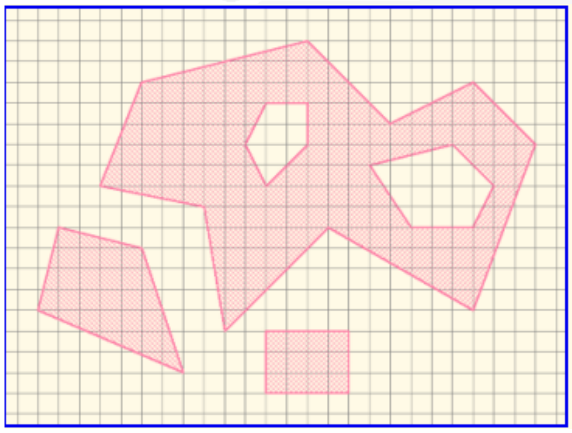
Abut: Candidate polygon abuts any filter polygon from outside or inside



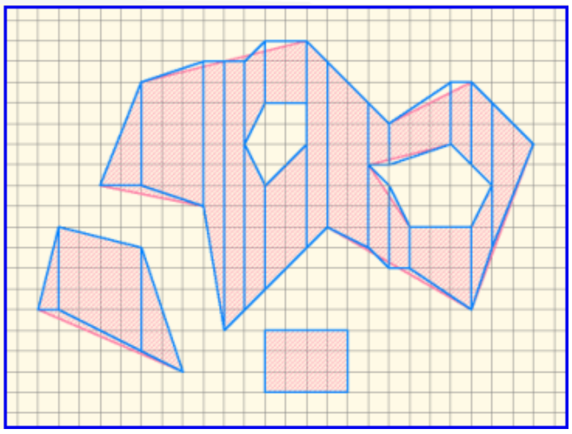
5. Trapezoidal Decomposition

The trapezoidal decomposition operation ("TRAPEZIFY") receives a polygon layer as its argument. The operation result is a set of polygons that is a trapezoidal decomposition of the input polygon layer.

Input polygon layer (red)



Result of trapezoidal decomposition (blue)



2.2 Input data

GPP does not include a loader for data from standard file formats like OASIS and the Shapefile format. Input and output data are pointers to a set of edges of polygons, where each edge is associated with the following information:

- 2 vertices (fixed-point integers, 31 or 53 bits max, input/output)
- a polygon ID (input)
- a layer ID (input)
- a winding count number (output of single-layer boolean operations)
- a layer bitmask (output of multiple-layer boolean operations)

Also, GPP supports all-angle geometry, not just Manhattan or 45-degree geometry.

The GPP specification is evolving and more input and output data members will appear as new functionality is added.

There is no external limit on input data size, as long as it fits in memory.

2.3 System Requirements

CPU/GPU:

The CPU version is written in pure C/C++ with OpenMP. It should work on all systems that are compliant with C/C++ and OpenMP. The GPU version requires an NVIDIA GPU with Compute Capability 1.3 or higher. (Note: earlier versions lack support for double precision, thus we cannot guarantee robustness on earlier versions.)

Development Tools:

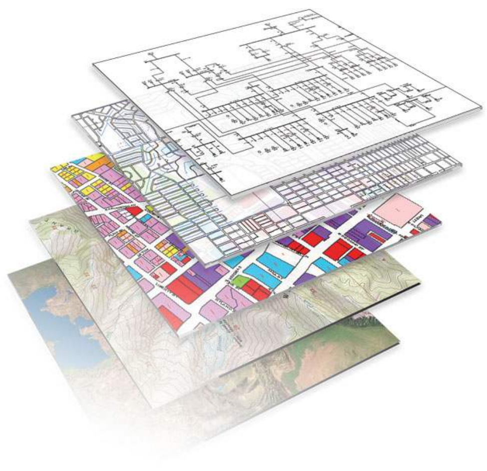
If compiling from source, GCC 4.3 (or higher) and CUDA 3.2 (or higher) are recommended.

OS:

Linux (32bit/64bit)

Microsoft Windows support is planned, but not currently available.

3 Performance



One of the target areas for GPP is geographic information systems (GIS). In GIS, a computational geometry engine is used in superimposing multiple different location data sets. Current GIS solutions rely on deployments that can only operate in narrow processor configurations. Yet user demands for larger datasets, spatial databases, and high-precision analysis require ever more computational power. GPP meets that challenge with a cutting edge algorithm that is fully optimized for multi-core/many core processor environments.

In the GIS industry, a de-facto standard software is "ArcGIS," developed by ESRI. We measured the elapsed time for GPP and ArcGIS to process two freely available datasets:

- Admin 1 - States, Provinces (14 MB)
- Urban Areas (12.5MB)

  (Downloaded from: http://www.naturalearthdata.com/downloads/10m-cultural-vectors/)

Fig 4 shows the results of a "boolean intersect operation" by three different engines. ArcGIS does not support multi-core processing. GPP achieved 7 times faster performance for the CPU version, and 18 times faster performance for the GPU version. For this evaluation, the CPU is an Intel Core i7 4Core @3.0Hz and the GPU is an NVIDIA GeForce GTX 680.

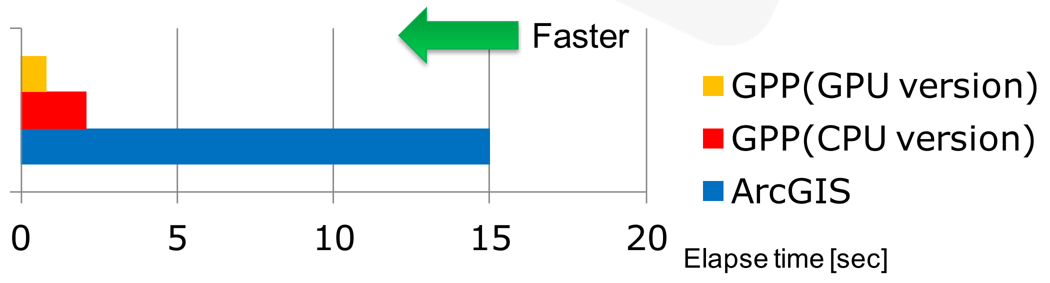


Fig4: Performance comparison to ArcGIS

4 Resources

These websites provide useful references to supplement the information contained in this paper:

▪ Wikipedia	http://en.wikipedia.org/wiki/Computational_geometry
▪ ArcGIS	http://www.arcgis.com/about/
▪ About Fixstars	http://www.fixstars.com/