

Advanced Programming Techniques Applied to:

CGAL's Arrangement Package

Ron Wein, Efi Fogel, Baruch Zukerman, and Dan Halperin
Tel Aviv Univ.

LCSD, San Diego
October 16, 2005



Overview

- Related Work
- CGAL
- Arrangements
- Adapters
- Decorators
- Observers
- Visitors
- Video



Overview

- Related Work
- CGAL
- Arrangements
- Adapters
- Decorators
- Observers
- Visitors
- Video



Related Work

- Keyser et al. implemented a module that constructs arrangements of algebraic curves constrained to some general-position assumptions
- The LEDA library contains a module that constructs and maintains arrangements of line segments
- CGAL's arrangement package constructs and maintains arrangements of arbitrary curves and supports operations and queries on such arrangements
 - All inputs are handled correctly (including degenerate)
 - Exact number types are used to achieve exact results



Overview

- Related Work
- **CGAL**
- Arrangements
- Adapters
- Decorators
- Observers
- Visitors
- Video



CGAL

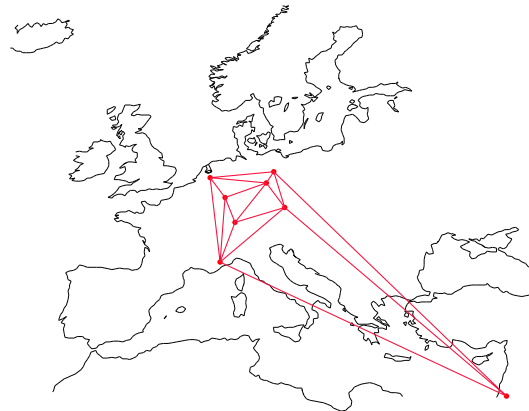
Written in C++

Follows the *generic programming* paradigm

Development started in 1995

Consortium of 6 active European sites:

- ❶ Utrecht University
- ❷ INRIA Sophia Antipolis
- ❸ ETH Zürich
- ❹ MPII Saarbrücken
- ❺ **Tel Aviv University**
- ❻ Freie Universität Berlin
- ❼ RISC Linz
- ❽ Martin-Luther-Universität Halle



CGAL Structure

Basic Library

Algorithms and Data Structures

Kernel

Geometric Objects of constant size
Geometric Operations on objects of constant size

Support Library

configurations, assertions,...

Visualization

File

I/O

Number Types

Generators

...



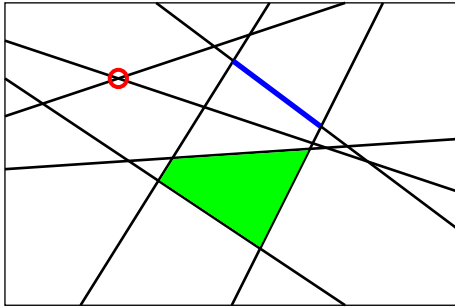
Overview

- Related Work
- CGAL
- Arrangements
- Adapters
- Decorators
- Observers
- Visitors
- Video

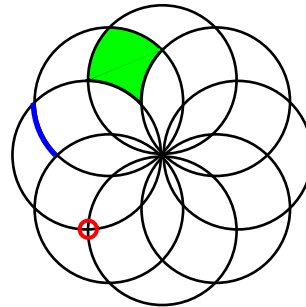


Arrangements

Given a collection Γ of planar curves, the arrangement $\mathcal{A}(\Gamma)$ is the partition of the plane into **vertices**, **edges** and **faces** induced by the curves of Γ



An arrangement of lines



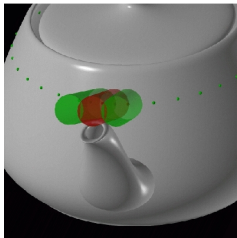
An arrangement of circles



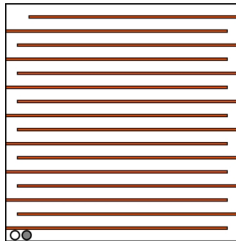
Arrangement Applications

Arrangements have numerous applications

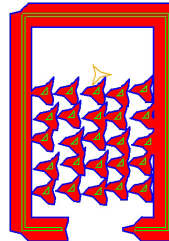
robot motion planning, computer vision, GIS, optimization, molecular biology, etc.



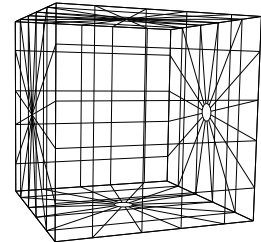
Path Verification
for NC-Machining



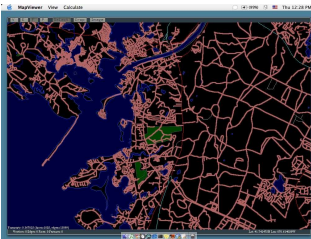
Hybrid Motion
Planning



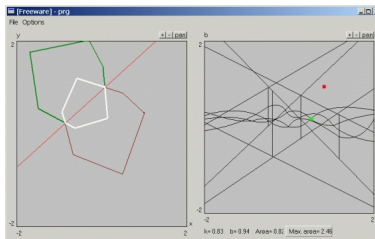
2D Minkowski
Sums



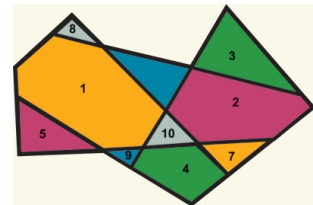
3D Minkowski
Sums



A planar map of the
Boston area



Maximizing the Area of an
Axis-Symmetric Polygon



Inner-cover of Non-
convex Shapes



Arrangement Goals

- Construct, maintain, modify, traverse, query and present subdivisions of the plane
- Robust, handles all degeneracies, exact
- Efficient
- **Generic**, easy to interface, extend, and adapt
- Part of the CGAL basic library



Arrangement Traits

- Separates geometric aspects from topological aspects
 - `Arrangement_2<Traits,Dcel>` — main component
- Parameter of package
 - Defines the family of curves of interest
 - Package can be used with any family of curves for which a traits class is supplied
- Aggregates
 - Geometric types (point, curve)
 - Operations over types (accessors, predicates, constructors)



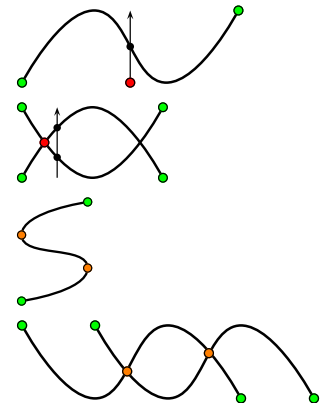
Arrangement Traits Requirements

- Types: `Curve_2`, `X_monotone_curve_2`, `Point_2`
- Methods:
 - ❶ Compare two points

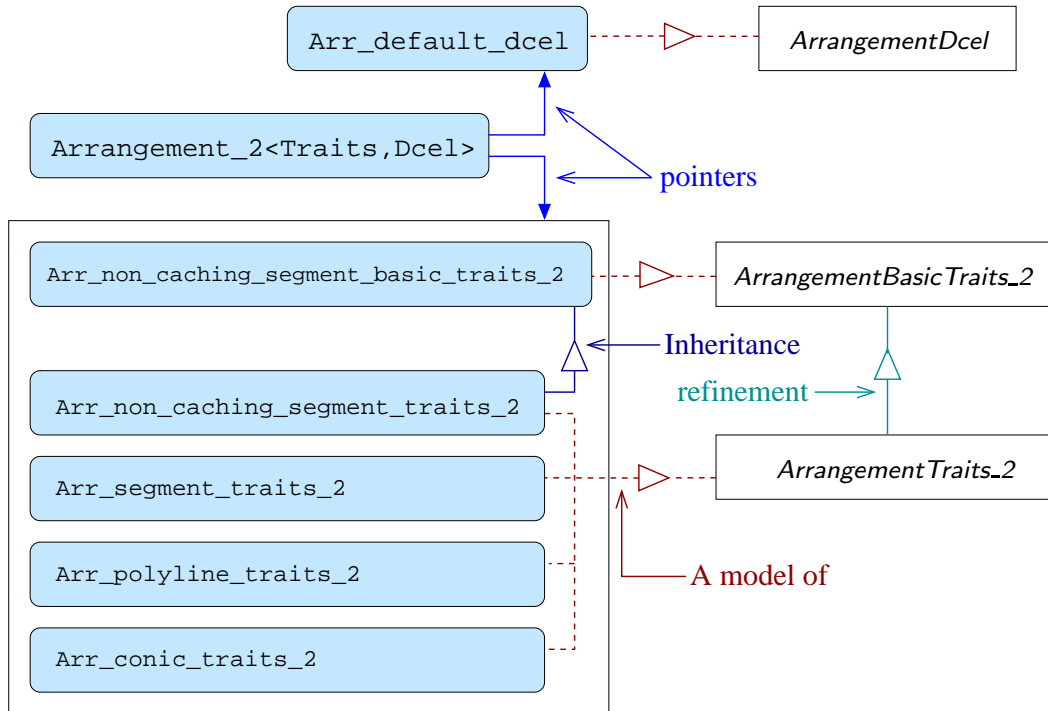


Arrangement Traits Requirements

- Types: `Curve_2`, `X_monotone_curve_2`, `Point_2`
- Methods:
 - ❶ Compare two points
 - ❷ Determine the relative position of a point and an x -monotone curve
 - ❸ Determine the relative position of two x -monotone curves to the left (or right) of a point
 - ❹ Subdivide a curve into x -monotone curves
 - ❺ Find all intersections of two x -monotone curves



Arrangement Architecture



Overview

- Related Work
- CGAL
- Arrangements
- **Adapters**
- Decorators
- Observers
- Visitors
- Video



Adapters

The *adapter* design-pattern “converts the interface of a class into another interface clients expect. Adapters let classes work together that could not otherwise, because of incompatible interfaces.” (Gamma *et al.*)



Adapters

The *adapter* design-pattern “converts the interface of a class into another interface clients expect. Adapters let classes work together that could not otherwise, because of incompatible interfaces.” (Gamma *et al.*)

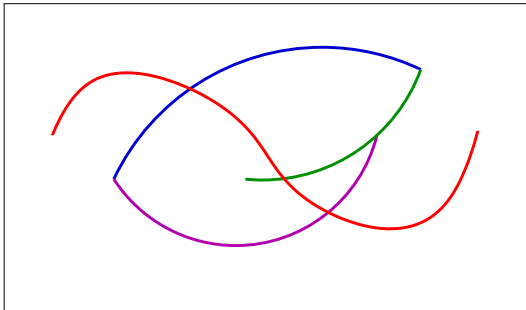
Adapters that appear in the arrangement package:

- The traits adapter
- The DCEL face extender
- Boost graph adapters

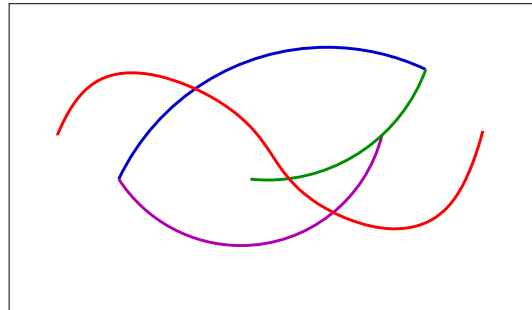


Boost Graph Adapters

- Boost Graph Library (BGL) supports graph algorithms
- Arrangements are embedded in the plane as planar graphs
 - ↳ Extend arrangements with the interface BGL expects



Primal adapter

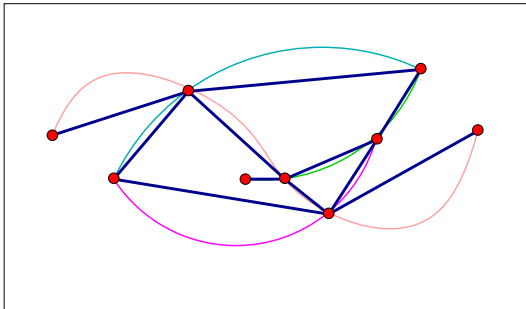


Dual adapter

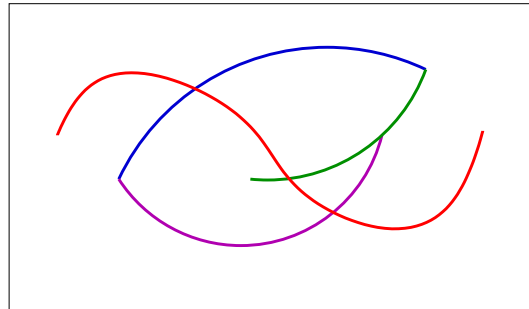


Boost Graph Adapters

- Boost Graph Library (BGL) supports graph algorithms
- Arrangements are embedded in the plane as planar graphs
 - ↳ Extend arrangements with the interface BGL expects



Primal adapter

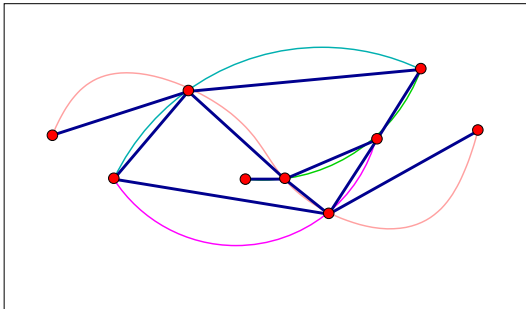


Dual adapter

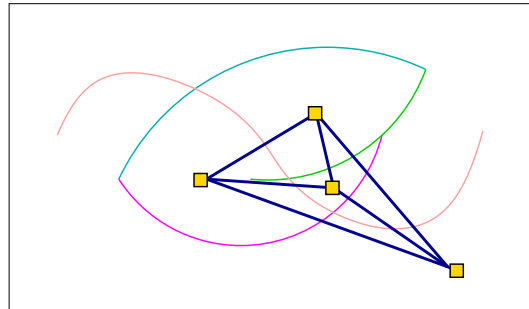


Boost Graph Adapters

- Boost Graph Library (BGL) supports graph algorithms
- Arrangements are embedded in the plane as planar graphs
 - ➔ Extend arrangements with the interface BGL expects



Primal adapter



Dual adapter



Overview

- Related Work
- CGAL
- Arrangements
- Adapters
- **Decorators**
- Observers
- Visitors
- Video



Decorators

The *decorator* design-pattern “*attaches additional responsibilities to an object dynamically. Decorators provide a flexible alternative to sub-classing for extending functionality.*”
(Gamma *et al.*)



Decorators

The *decorator* design-pattern “*attaches additional responsibilities to an object dynamically. Decorators provide a flexible alternative to sub-classing for extending functionality.*” (Gamma *et al.*)

Decorators that appear in the arrangement package:

- Model the *ArrangementTraits_2* concept
 - Merged curve-data traits — extends the *x*-monotone curve-type with a single data field
 - Consolidated curve-data traits — extends the *x*-monotone curve-type with a container of data fields
 - ◆ Used to implement the *Arrangement_with_history_2*
Maintains its construction history



Arrangement with History: the Netherlands



Roads, railroads, rivers and water canals on the map of the Netherlands.

Bridges are marked by circles.

- border — 13 polylines, 373 segments
- roads — 877 polylines, 2360 segments
- water routes — 88 polylines, 1397 segments



Overview

- Related Work
- CGAL
- Arrangements
- Adapters
- Decorators
- Observers
- Visitors
- Video



Observers

The *observer* design-pattern “*defines a one-to-many dependency between objects, so that when one object changes state, all its dependents are notified and updated automatically.*” (Gamma *et al.*)



Observers

The *observer* design-pattern “*defines a one-to-many dependency between objects, so that when one object changes state, all its dependents are notified and updated automatically.*” (Gamma *et al.*)

Observers used by the Arrangement package:

- Point-location strategies that maintain auxiliary data
 - *Landmark* point-location
 - Trapezoidal decomposition based point-location
- Dynamic maintenance of attributes stored at the arrangement features



Overview

- Related Work
- CGAL
- Arrangements
- Adapters
- Decorators
- Observers
- Visitors
- Video



Visitors

The *visitor* design-pattern “represents an operation to be performed on an object or on the elements of an object structure. Visitors allow the definition of new operations without changing the classes of the elements on which they operate.” (Gamma *et al.*)



Visitors

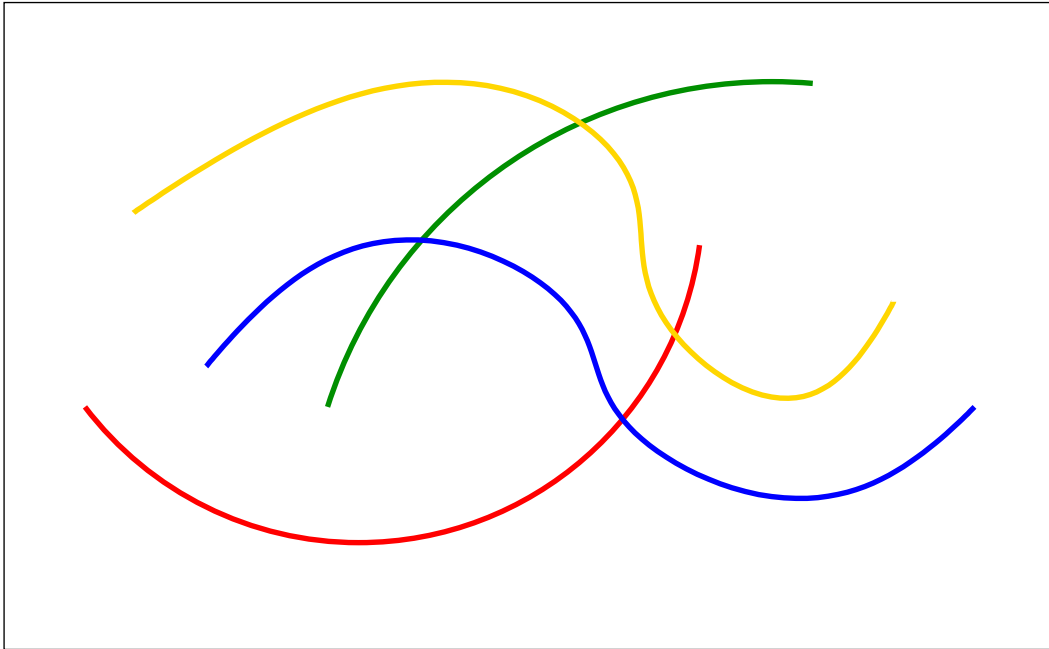
The *visitor* design-pattern “*represents an operation to be performed on an object or on the elements of an object structure. Visitors allow the definition of new operations without changing the classes of the elements on which they operate.*” (Gamma *et al.*)

Implementing an algorithm based on a framework below reduces to implementing an appropriate visitor class

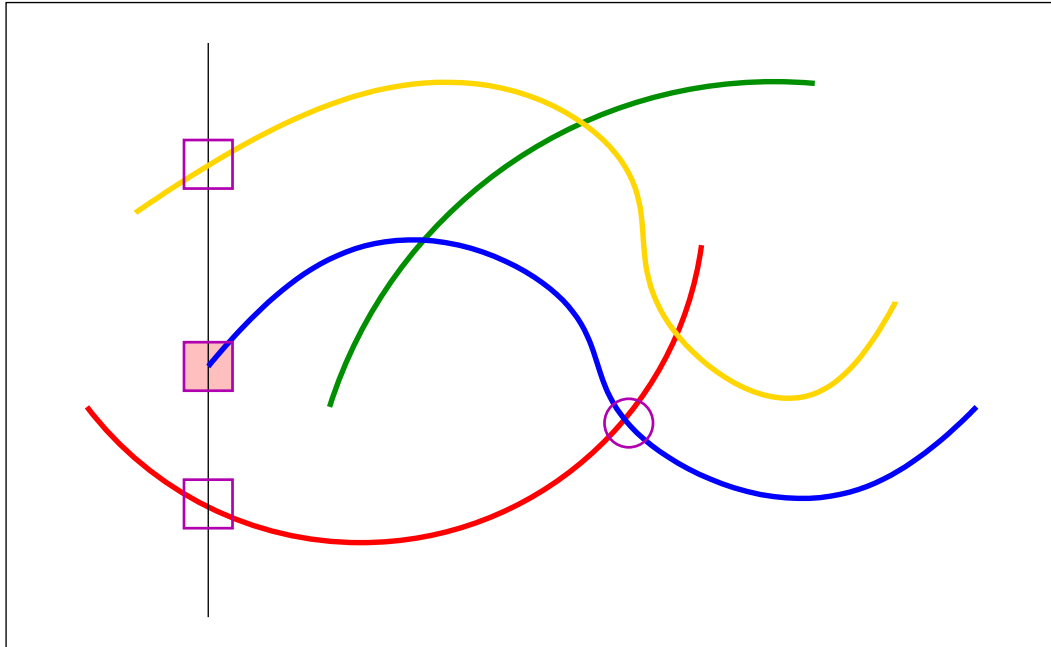
- (Vertical) line sweep of the plane — given a set of curves, identify all endpoints and intersection points
- Zone computation of a curve — given an arrangement and an x -monotone curve, identify all arrangement cells that the curve crosses



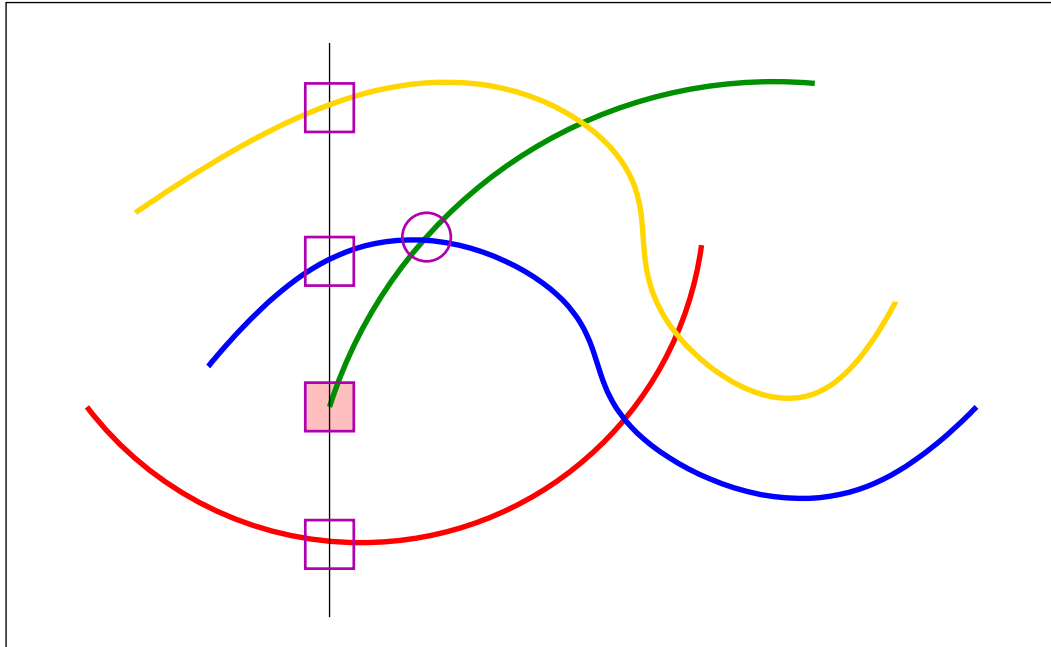
Sweep Line



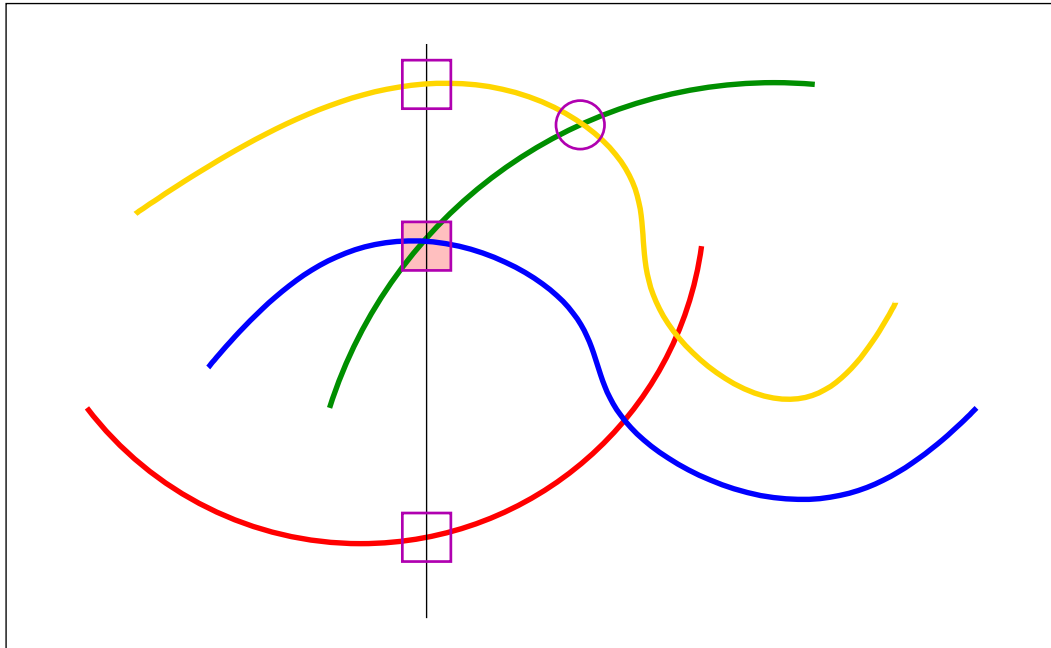
Sweep Line



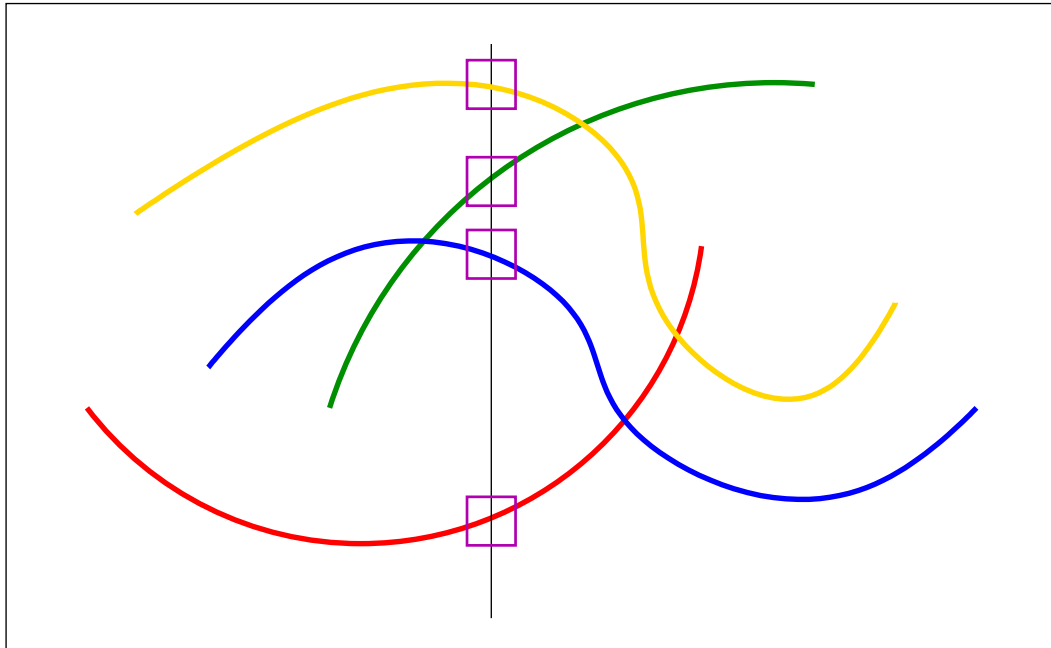
Sweep Line



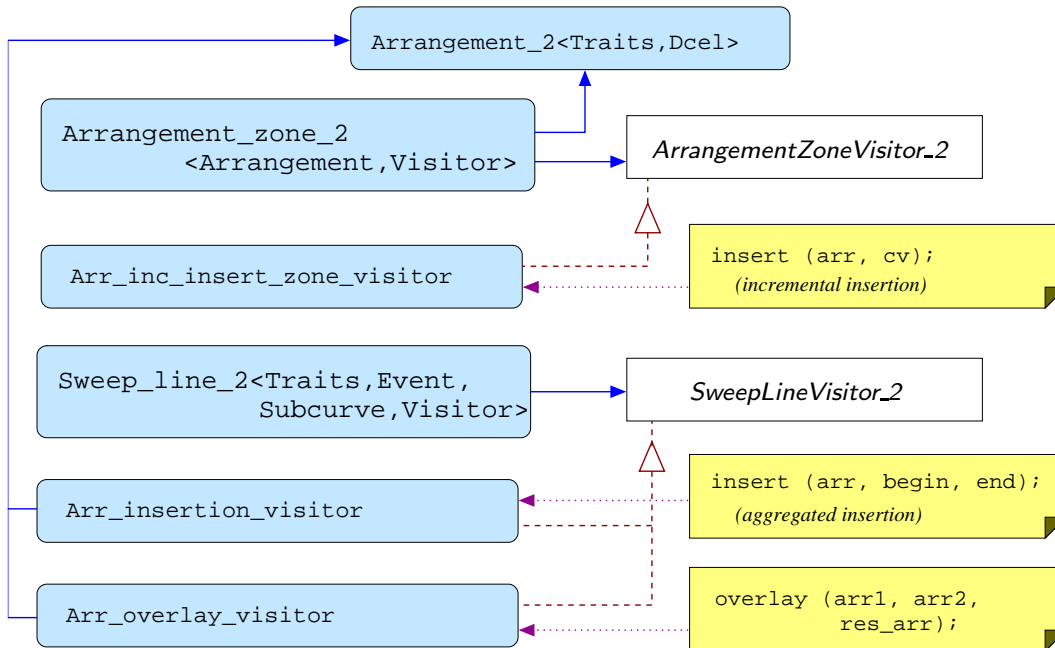
Sweep Line



Sweep Line



Free Functions



Overview

- Related Work
- CGAL
- Arrangements
- Adapters
- Decorators
- Observers
- Visitors
- Video



Video

`http://www.cs.tau.ac.il/~efif/CD/movies/Mink3d.mov`



TOC

Overview ❖

Overview ❖

Related Work ❖

Overview ❖

CGAL ❖

CGAL Structure ❖

Overview ❖

Arrangements ❖

Arrangement Applications ❖

Arrangement Goals ❖

Arrangement Traits ❖

Arrangement Traits Requirements ❖

Arrangement Architecture ❖

Overview ❖

Adapters ❖

Boost Graph Adapters ❖

Boost Graph Adapters ❖

Boost Graph Adapters ❖

Overview ❖

Decorators ❖

Arrangement with History: the Netherlands ❖

Overview ❖

Observers ❖

Notification Mechanism ❖

Overview ❖

Visitors ❖

Sweep Line ❖

Sweep Line ❖



[Sweep Line](#) ❖

[Sweep Line](#) ❖

[Sweep Line](#) ❖

[Free Functions](#) ❖

[Overview](#) ❖

[Video](#) ❖

