

Algebraic methods for geometric modeling

Julien Wintz



PhD defense, Sophia-Antipolis, May 5th 2008

Domain

- Solids
 - Constructive representation
 - Boundary representation
- Curves and surfaces
 - Linear representations
 - Non-linear representations

CSG

- Primitive solids
- Boolean operations
- Rigid motions

B-Rep

- Vertices
- Edges
- Faces

Domain

- Solids
 - Constructive representation
 - Boundary representation
- Curves and surfaces
 - Linear representations
 - Non-linear representations

Linear representations

- Meshes
- Inaccurate approximation
- Huge amount of data

Non-linear representations

- Parametric or implicit
- Accurate approximation
- Compact

Problem

- Perform boolean operations on curves and surfaces
- Describe resulting shapes by their boundary

Problem

- Perform boolean operations on curves and surfaces
- Describe resulting shapes by their boundary

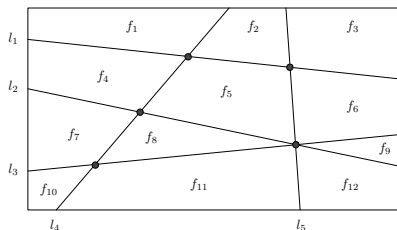


Figure: an arrangement of lines.

Problem






- Perform boolean operations on curves and surfaces
- Describe resulting shapes by their boundary
- Representation difficult to manipulate
 - Topology
 - Intersection
 - Self-intersection
- Need of algebraic tools
 - Polynomial solving
 - Algebraic numbers
 - Resultants

Problem

- Perform boolean operations on curves and surfaces
- Describe resulting shapes by their boundary
- Representation difficult to manipulate
 - Topology
 - Intersection
 - Self-intersection
- Need of algebraic tools
 - Polynomial solving
 - Algebraic numbers
 - Resultants

Previous work

Bibliography

-  G.E. Collins. *Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition.* 1976.
-  J.L. Bentley and T.A. Ottmann. *Algorithms for reporting and counting geometric intersections.* 1979.
-
-  L. Gonzalez-Vega and I. Necula. *Efficient topology determination of [...] algebraic plane curves.* 2002.
-  V. Milenkovic and E. Sacks. *An approximate arrangement algorithm for semi-algebraic curves.* 2006.
-  Iddo Hanniel and Ron Wein. *Computation of Arrangements of Bezier Curves.* 2007.

Sweep






- Focuses on critical points
- Implies projection
- Non adaptive scheme
- Does not easily extend to higher dimension

Subdivision

- Detects critical points
- Avoids projection
- Adaptive scheme
- Extends easily to higher dimension

Previous work

Bibliography

-  G.E. Collins. *Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition.* 1976.
-  J.L. Bentley and T.A. Ottmann. *Algorithms for reporting and counting geometric intersections.* 1979.
.....
-  L. Gonzalez-Vega and I. Necula. *Efficient topology determination of [...] algebraic plane curves.* 2002.
-  V. Milenkovic and E. Sacks. *An approximate arrangement algorithm for semi-algebraic curves.* 2006.
-  Iddo Hanniel and Ron Wein. *Computation of Arrangements of Bezier Curves.* 2007.

Sweep

- Focuses on critical points
- Implies projection
- Non adaptive scheme
- Does not easily extend to higher dimension

Subdivision

- Detects critical points
- Avoids projection
- Adaptive scheme
- Extends easily to higher dimension

Approach

- Generic algorithm
 - Many dimensions
 - Heterogeneous
 - Static or dynamic
 - Subdivision scheme
- Abstract algorithm
 - Generic algorithm produces local problems
 - Isolation: Subdivision criteria
 - Solving: Local treatment

Validation

- Specialization for curves
 - Non singular configurations
 - Singular configurations
- Specialization for surfaces
 - 2-dimensional strata
 - 1-dimensional strata
 - 0-dimensional strata
- Implementation within Axel

Outline

- ① Generic algorithm
 - Computing regions
 - Locating conflicts
 - Updating regions
- ② Specialization for curves
 - Implicit curves
 - Parametric curves
- ③ Specialization for surfaces
 - Implicit surfaces
 - Parametric surfaces
- ④ An algebraic geometric modeler
 - User perspective
 - Developer perspective

Outline

- ① Generic algorithm
 - Computing regions
 - Locating conflicts
 - Updating regions
- ② Specialization for curves
 - Implicit curves
 - Parametric curves
- ③ Specialization for surfaces
 - Implicit surfaces
 - Parametric surfaces
- ④ An algebraic geometric modeler
 - User perspective
 - Developer perspective

A generic incremental randomized dynamic algorithm

generic: works for different natures of objects.

incremental: objects are processed one by one.

randomized: data are inserted in random order.

semi-dynamic: new objects can be inserted.

A generic incremental randomized dynamic algorithm

generic: works for different natures of objects.

incremental: objects are processed one by one.

randomized: data are inserted in random order.

semi-dynamic: new objects can be inserted.

A generic incremental randomized dynamic algorithm

generic: works for different natures of objects.

incremental: objects are processed one by one.

randomized: data are inserted in random order.

semi-dynamic: new objects can be inserted.

A generic incremental randomized dynamic algorithm

generic: works for different natures of objects.

incremental: objects are processed one by one.

randomized: data are inserted in random order.

semi-dynamic: new objects can be inserted.

Outline

- ① **Generic algorithm**
 - Computing regions
 - Locating conflicts
 - Updating regions
- ② Specialization for curves
 - Implicit curves
 - Parametric curves
- ③ Specialization for surfaces
 - Implicit surfaces
 - Parametric surfaces
- ④ An algebraic geometric modeler
 - User perspective
 - Developer perspective

Subdivision

Input: an object o

Output: a list of regions

create an quadtree Q ;

create a list of cells \mathcal{C} ;

$\mathcal{C} \leftarrow \text{root}(Q)$;

while $\mathcal{C} \neq \emptyset$ **do**

$c = \text{pop}(\mathcal{C})$;

if $\text{regular}(o, c)$ **then**

$Q \leftarrow \text{topology}(o, c)$;

else

$\mathcal{C} \leftarrow \text{subdivide}(o, c)$;

end

end

return $\text{fusion}(Q)$;

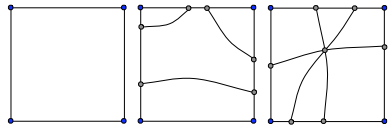


Figure: Regular cells.

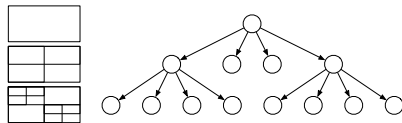


Figure: Subdivision algorithm.

Subdivision

The following operations remain to be clarified:

regularity: the *specific* operation which checks if regions can be computed from an object within a cell of the subdivision, i.e. if the object is regular in the cell.

subdivide: the *generic* operation which subdivides a cell into four children, saving computation effort.

topology: the *specific* operation which computes regions in a regular cell.

fusion: the *generic* operation which merges regions stored in each leaf node of the tree.

Subdivision

The following operations remain to be clarified:

regularity: the *specific* operation which checks if regions can be computed from an object within a cell of the subdivision, i.e. if the object is regular in the cell.

subdivide: the *generic* operation which subdivides a cell into four children, saving computation effort.

topology: the *specific* operation which computes regions in a regular cell.

fusion: the *generic* operation which merges regions stored in each leaf node of the tree.

Subdivision

The following operations remain to be clarified:

regularity: the *specific* operation which checks if regions can be computed from an object within a cell of the subdivision, i.e. if the object is regular in the cell.

subdivide: the *generic* operation which subdivides a cell into four children, saving computation effort.

topology: the *specific* operation which computes regions in a regular cell.

fusion: the *generic* operation which merges regions stored in each leaf node of the tree.

Subdivision

The following operations remain to be clarified:

regularity: the *specific* operation which checks if regions can be computed from an object within a cell of the subdivision, i.e. if the object is regular in the cell.

subdivide: the *generic* operation which subdivides a cell into four children, saving computation effort.

topology: the *specific* operation which computes regions in a regular cell.

fusion: the *generic* operation which merges regions stored in each leaf node of the tree.

Subdivision

The following operations remain to be clarified:

regularity: the *specific* operation which checks if regions can be computed from an object within a cell of the subdivision, i.e. if the object is regular in the cell.

subdivide: the *generic* operation which subdivides a cell into four children, saving computation effort.

topology: the *specific* operation which computes regions in a regular cell.

fusion: the *generic* operation which merges regions stored in each leaf node of the tree.

Regularity

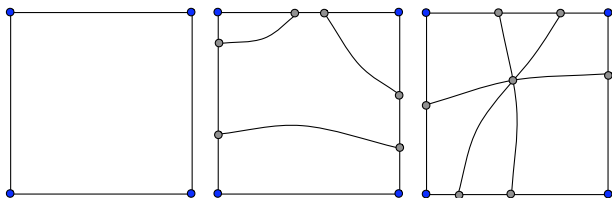
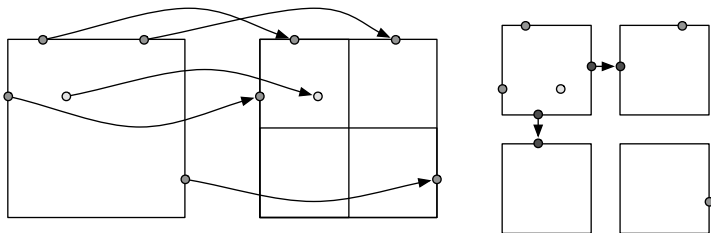


Figure: Allowed configurations of a cell of subdivision.

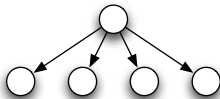
Given the representation of an object:

- compute the intersection of the object with the cell
- define and compute $(x|y)$ -critical points
- define and compute *singular* points

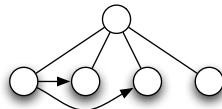
Subdivide



A subdivision cell
in a quadtree



1. Vertical inheritance



2. Horizontal inheritance

Figure: Inheriting computed information.

Fusion

Input: a node n of the quadtree

Output: a list of regions

if isLeaf(n) **then**
 return merge(n) ;

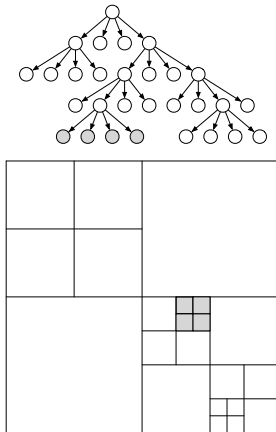
else

$\mathcal{L}_1 = \text{fusion}(n.ne) \cup$
 fusion($n.nw$) ;

$\mathcal{L}_2 = \text{fusion}(n.se) \cup$
 fusion($n.sw$) ;

return merge($\mathcal{L}_1 \cup \mathcal{L}_2$) ;

end



Fusion

Input: a node n of the quadtree

Output: a list of regions

if isLeaf(n) **then**
 return merge(n) ;

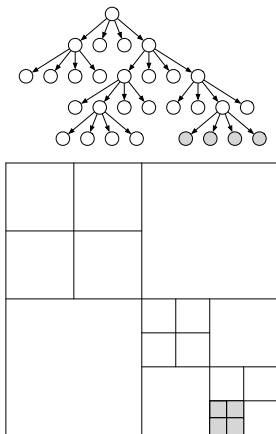
else

$\mathcal{L}_1 = \text{fusion}(n.ne) \cup$
 fusion($n.nw$) ;

$\mathcal{L}_2 = \text{fusion}(n.se) \cup$
 fusion($n.sw$) ;

return merge($\mathcal{L}_1 \cup \mathcal{L}_2$) ;

end



Fusion

Input: a node n of the quadtree

Output: a list of regions

if $\text{isLeaf}(n)$ **then**
 return $\text{merge}(n)$;

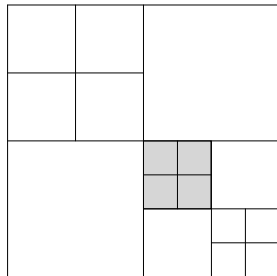
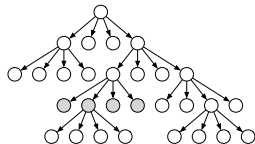
else

$\mathcal{L}_1 = \text{fusion}(n.ne) \cup$
 $\text{fusion}(n.nw)$;

$\mathcal{L}_2 = \text{fusion}(n.se) \cup$
 $\text{fusion}(n.sw)$;

return $\text{merge}(\mathcal{L}_1 \cup \mathcal{L}_2)$;

end



Fusion

Input: a node n of the quadtree

Output: a list of regions

if `isLeaf(n)` **then**
 return `merge(n)` ;

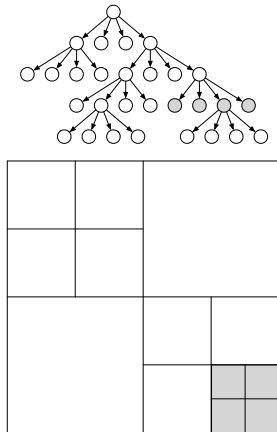
else

$\mathcal{L}_1 = \text{fusion}(n.ne) \cup$
 $\text{fusion}(n.nw)$;

$\mathcal{L}_2 = \text{fusion}(n.se) \cup$
 $\text{fusion}(n.sw)$;

return `merge($\mathcal{L}_1 \cup \mathcal{L}_2$)` ;

end



Fusion

Input: a node n of the quadtree

Output: a list of regions

if isLeaf(n) **then**
 return merge(n) ;

else

$\mathcal{L}_1 = \text{fusion}(n.ne) \cup$

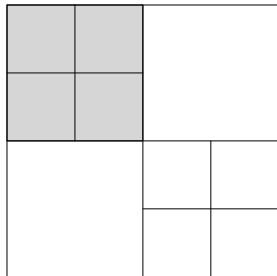
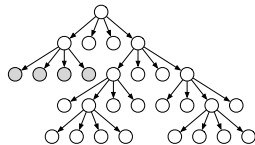
 fusion($n.nw$) ;

$\mathcal{L}_2 = \text{fusion}(n.se) \cup$

 fusion($n.sw$) ;

return merge($\mathcal{L}_1 \cup \mathcal{L}_2$) ;

end



Fusion

Input: a node n of the quadtree

Output: a list of regions

if isLeaf(n) **then**
 return merge(n) ;

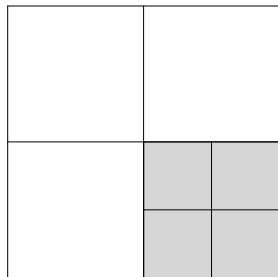
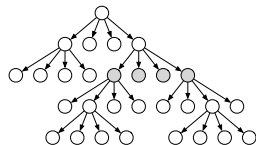
else

$\mathcal{L}_1 = \text{fusion}(n.ne) \cup$
 fusion($n.nw$) ;

$\mathcal{L}_2 = \text{fusion}(n.se) \cup$
 fusion($n.sw$) ;

return merge($\mathcal{L}_1 \cup \mathcal{L}_2$) ;

end



Fusion

Input: a node n of the quadtree

Output: a list of regions

if `isLeaf(n)` **then**
 return `merge(n)` ;

else

$\mathcal{L}_1 = \text{fusion}(n.ne) \cup$

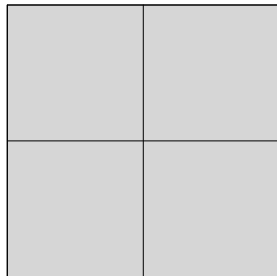
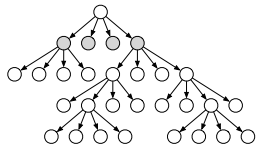
`fusion($n.nw$)` ;

$\mathcal{L}_2 = \text{fusion}(n.se) \cup$

`fusion($n.sw$)` ;

return `merge($\mathcal{L}_1 \cup \mathcal{L}_2$)` ;

end



Fusion

Input: a node n of the quadtree

Output: a list of regions

if `isLeaf(n)` **then**
 return `merge(n)` ;

else

$\mathcal{L}_1 = \text{fusion}(n.ne) \cup$

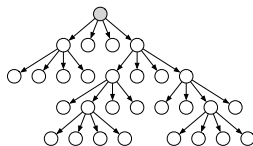
`fusion($n.nw$)` ;

$\mathcal{L}_2 = \text{fusion}(n.se) \cup$

`fusion($n.sw$)` ;

return `merge($\mathcal{L}_1 \cup \mathcal{L}_2$)` ;

end



Outline

① Generic algorithm

Computing regions

Locating conflicts

Updating regions

② Specialization for curves

Implicit curves

Parametric curves

③ Specialization for surfaces

Implicit surfaces

Parametric surfaces

④ An algebraic geometric modeler

User perspective

Developer perspective

The region segmentation

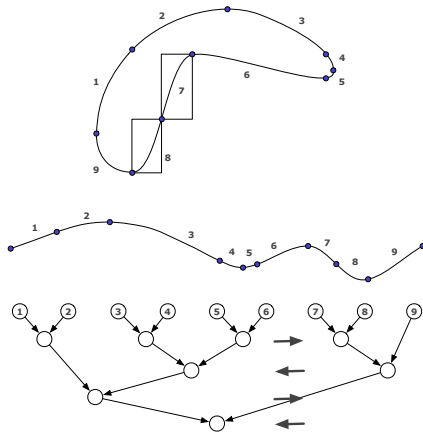


Figure: Building the region segmentation.

Querying the segmentations

Input: two segmentation nodes n_1 and n_2

Output: a list of bounding boxes
create a list of bounding boxes \mathcal{L} ;

if !intersects(n_1, n_2) **then**
 return \mathcal{L} ;

end

if isLeaf(n_1) **and** isLeaf(n_2) **then**
 $\mathcal{L} \ll \text{intersect}(n_1, n_2)$;

else if isLeaf(n_1) **and** !isLeaf(n_2) **then**
 $\mathcal{L} \ll \text{query}(n_1, \text{left}(n_2))$;
 $\mathcal{L} \ll \text{query}(n_1, \text{right}(n_2))$;

else if !isLeaf(n_1) **and** isLeaf(n_2) **then**
 $\mathcal{L} \ll \text{query}(\text{left}(n_1), n_2)$;
 $\mathcal{L} \ll \text{query}(\text{right}(n_1), n_2)$;

else

$\mathcal{L} \ll \text{query}(\text{left}(n_1), \text{left}(n_2))$;
 $\mathcal{L} \ll \text{query}(\text{left}(n_1), \text{right}(n_2))$;
 $\mathcal{L} \ll \text{query}(\text{right}(n_1), \text{left}(n_2))$;
 $\mathcal{L} \ll \text{query}(\text{right}(n_1), \text{right}(n_2))$;

return \mathcal{L} ;

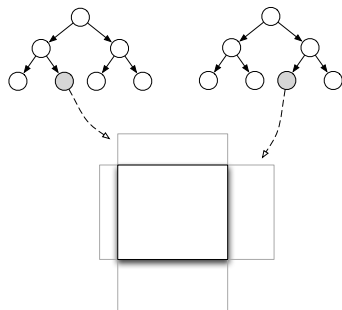


Figure: The segmentation query.

Outline

① Generic algorithm

Computing regions

Locating conflicts

Updating regions

② Specialization for curves

Implicit curves

Parametric curves

③ Specialization for surfaces

Implicit surfaces

Parametric surfaces

④ An algebraic geometric modeler

User perspective

Developer perspective

Conflicting regions

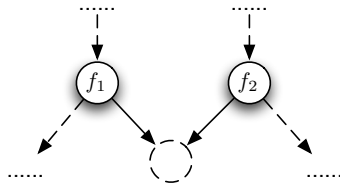
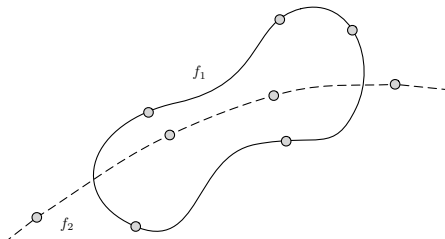


Figure: Updating the augmented influence graph.

- Resolved by computing the intersection of conflicting regions
- Append it to the graph, as a child of corresponding nodes

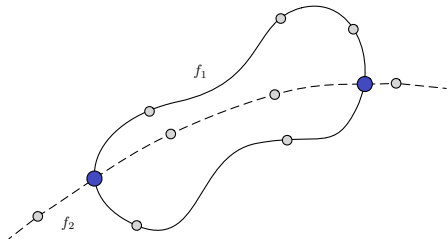
A generic boolean operation

- Walk-about on the border of regions
- Computing intersection points
- Computing navigation information
- Computing the resulting region



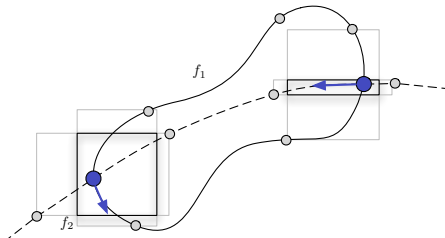
A generic boolean operation

- Walk-about on the border of regions
- Computing intersection points
- Computing navigation information
- Computing the resulting region



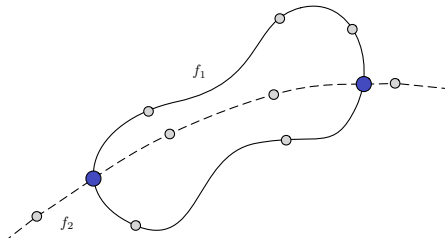
A generic boolean operation

- Walk-about on the border of regions
- Computing intersection points
- Computing navigation information
- Computing the resulting region



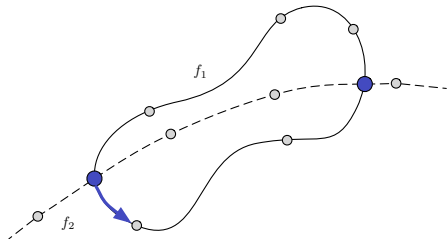
A generic boolean operation

- Walk-about on the border of regions
- Computing intersection points
- Computing navigation information
- Computing the resulting region



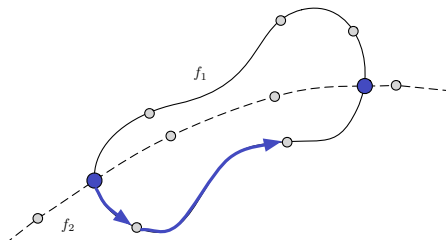
A generic boolean operation

- Walk-about on the border of regions
- Computing intersection points
- Computing navigation information
- Computing the resulting region



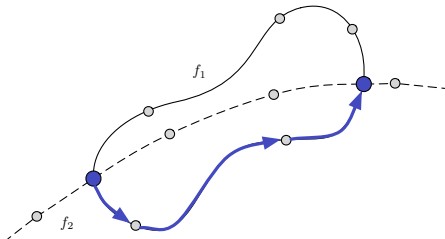
A generic boolean operation

- Walk-about on the border of regions
- Computing intersection points
- Computing navigation information
- Computing the resulting region



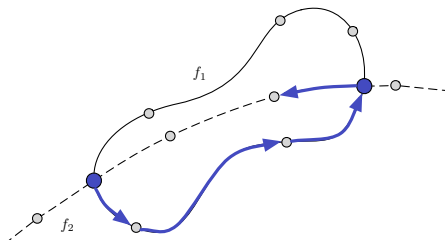
A generic boolean operation

- Walk-about on the border of regions
- Computing intersection points
- Computing navigation information
- Computing the resulting region



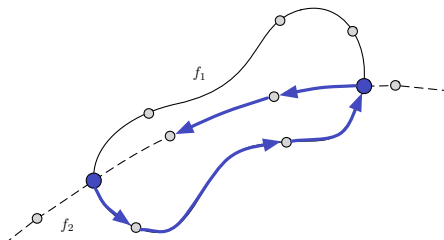
A generic boolean operation

- Walk-about on the border of regions
- Computing intersection points
- Computing navigation information
- Computing the resulting region



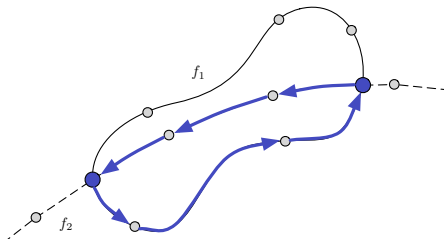
A generic boolean operation

- Walk-about on the border of regions
- Computing intersection points
- Computing navigation information
- Computing the resulting region



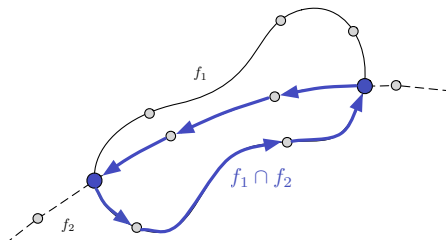
A generic boolean operation

- Walk-about on the border of regions
- Computing intersection points
- Computing navigation information
- Computing the resulting region



A generic boolean operation

- Walk-about on the border of regions
- Computing intersection points
- Computing navigation information
- Computing the resulting region



Generic algorithm

Specialization for curves

Specialization for surfaces

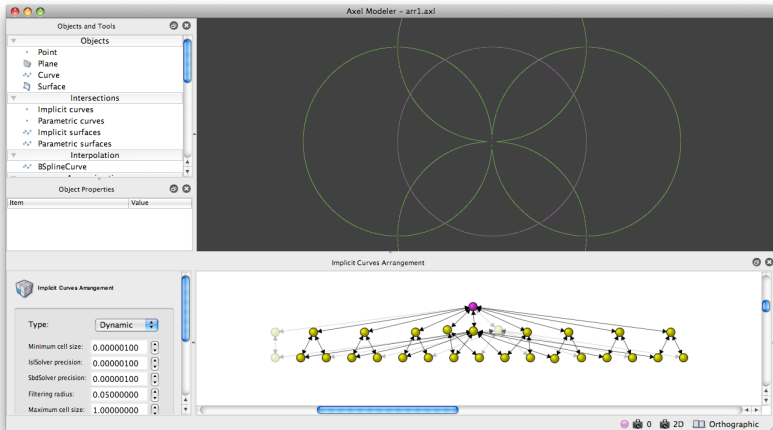
An algebraic geometric modeler

Computing regions

Locating conflicts

Updating regions

Demonstration



Outline

- ① Generic algorithm
 - Computing regions
 - Locating conflicts
 - Updating regions
- ② Specialization for curves
 - Implicit curves
 - Parametric curves
- ③ Specialization for surfaces
 - Implicit surfaces
 - Parametric surfaces
- ④ An algebraic geometric modeler
 - User perspective
 - Developer perspective

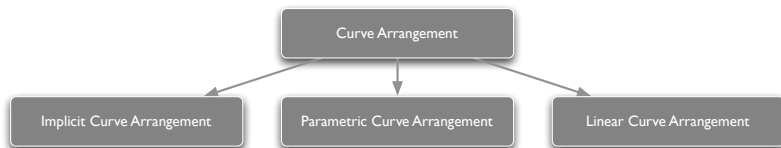
Specialization

Generic algorithm

- Global problem
- Any representation

Specific algorithm

- Local problem
- One representation



Specialization

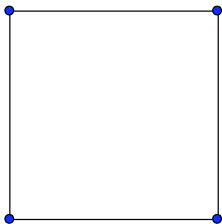
Characterize the situation:

- Empty cells
- Non-singular cells
- Singular cells

Specialization

Characterize the situation:

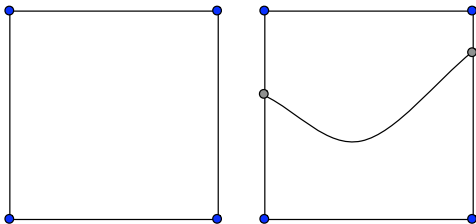
- Empty cells
- Non-singular cells
- Singular cells



Specialization

Characterize the situation:

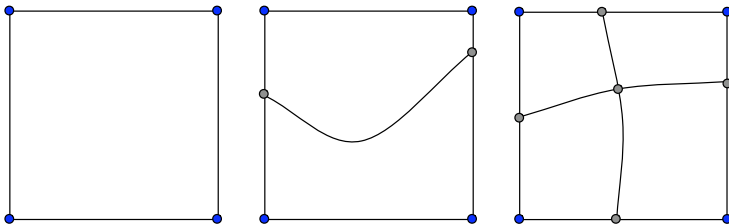
- Empty cells
- Non-singular cells
- Singular cells



Specialization

Characterize the situation:

- Empty cells
- Non-singular cells
- Singular cells



Outline

- ① Generic algorithm
 - Computing regions
 - Locating conflicts
 - Updating regions
- ② Specialization for curves
 - Implicit curves**
 - Parametric curves
- ③ Specialization for surfaces
 - Implicit surfaces
 - Parametric surfaces
- ④ An algebraic geometric modeler
 - User perspective
 - Developer perspective

Definitions

Monomial basis

We denote by $f_k(x, y) \in \mathbb{R}[x, y]$ the polynomial defining the implicit curve corresponding to the object o_k .

Bernstein basis

The specific operations will be performed on the Bernstein representation of f_k on $D = [a, b] \times [c, d]$:

$$f_k(x, y) = \sum_{i=0}^{d_{x,k}} \sum_{j=0}^{d_{y,k}} b_{i,j}^k B_{d_{x,k}}^i(x; a, b) B_{d_{y,k}}^j(y; c, d),$$

where $B_d^i(x; u, v) = \binom{d}{i} (x - u)^i (v - x)^{d-i} (v - u)^{-d}$.

Definitions

Critical points

Solve with multivariate subdivision solver:

$$\left\{ \begin{array}{l} f_k(x, y) = 0 \\ \partial_x f_k(x, y) = 0 \end{array} \right. \text{ and } \left\{ \begin{array}{l} f_k(x, y) = 0 \\ \partial_y f_k(x, y) = 0 \end{array} \right.$$

Singular points

Solve with multivariate subdivision solver:

$$\left\{ \begin{array}{l} f_k(x, y) = 0 \\ \partial_x f_k(x, y) = 0 \\ \partial_y f_k(x, y) = 0 \end{array} \right.$$

Definitions

Curve-Cell intersection points

Solve with univariate subdivision solver:

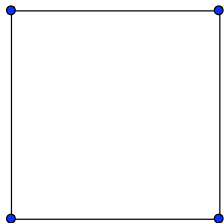
- $f(x, y_{min})$ and $f(x, y_{max})$ in x .
- $f(x_{min}, y)$ and $f(x_{max}, y)$ in y .

Curve-Curve intersection points

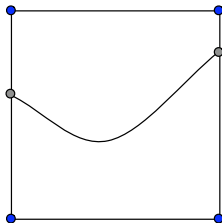
Solve with multivariate subdivision solver:

$$\begin{cases} f_k(x, y) = 0 \\ f_l(x, y) = 0 \end{cases}$$

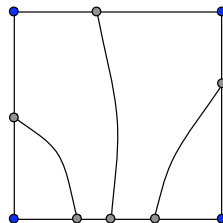
Regularity criteria: non singular cells



(a) Empty cell



(b) One branch



(c) Many branches

Regularity criteria: non singular cells

Definition

A domain \mathcal{D} is *x-regular* (resp. *y-regular*) for \mathcal{C} if \mathcal{C} is smooth in \mathcal{D} and it has no vertical (resp. horizontal) tangents. This is algebraically formulated as the following condition:

$$\mathcal{Z}(f, \partial_y f) \cap \mathcal{D} = \emptyset \text{ (resp. } \mathcal{Z}(f, \partial_x f) \cap \mathcal{D} = \emptyset).$$

Proposition

Let $\mathcal{C} = \mathcal{Z}(f)$. If \mathcal{D} is a *x-regular* domain, the topology of \mathcal{C} in \mathcal{D} is uniquely determined by its intersection $\mathcal{C} \cap \partial\mathcal{D}$ with the boundary of \mathcal{D} .

Regularity criteria: non singular cells

Definition

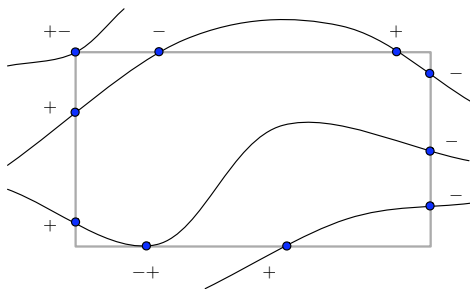
For a point $p \in \mathcal{C} \cap \partial\mathcal{D}$, we define its x -index.

- + if \mathcal{C} enters \mathcal{D} locally: there exists a local left (resp. right) tangent lying outside (resp. inside) \mathcal{D} .
- if \mathcal{C} exits \mathcal{D} locally: there exists a local left (resp. right) tangent lying inside (resp. outside) \mathcal{D} .
- + - if \mathcal{C} is tangent to \mathcal{D} and does not enter it locally: $\mathcal{C} - \{p\}$ locally lies outside \mathcal{C} .
- + if \mathcal{C} is tangent to \mathcal{D} and does not exit it locally: $\mathcal{C} \subset \mathcal{D}$.

Regularity criteria: non singular cells

Lemma

If \mathcal{C} is x -regular in \mathcal{D} , then a branch of $\mathcal{C} \cap \mathcal{D}$ connects a point p of x -index $+$ to a point q of x -index $-$, such that $x_p < x_q$.



Topology: non singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$\text{index}(\mathcal{C} \cap \partial\mathcal{D})$;

$\mathcal{L} \ll \text{order}(\mathcal{C} \cap \partial\mathcal{D})$;

while $\mathcal{L} \neq \emptyset$ **do**

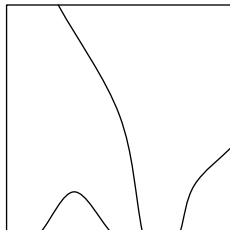
 choose (p, q) ;

$\mathcal{B} \leftarrow \mathfrak{b} = [p, q]$;

$\mathcal{L} = \mathcal{L} \setminus p$;

$\mathcal{L} = \mathcal{L} \setminus q$;

end



Topology: non singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$\text{index}(\mathcal{C} \cap \partial\mathcal{D})$;

$\mathcal{L} \ll \text{order}(\mathcal{C} \cap \partial\mathcal{D})$;

while $\mathcal{L} \neq \emptyset$ **do**

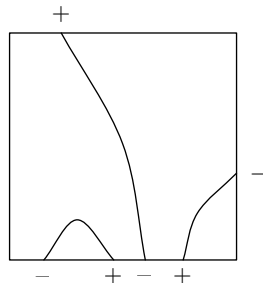
 choose (p, q) ;

$\mathcal{B} \leftarrow \mathcal{b} = [p, q]$;

$\mathcal{L} = \mathcal{L} \setminus p$;

$\mathcal{L} = \mathcal{L} \setminus q$;

end



Topology: non singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$\text{index}(\mathcal{C} \cap \partial\mathcal{D})$;

$\mathcal{L} \ll \text{order}(\mathcal{C} \cap \partial\mathcal{D})$;

while $\mathcal{L} \neq \emptyset$ **do**

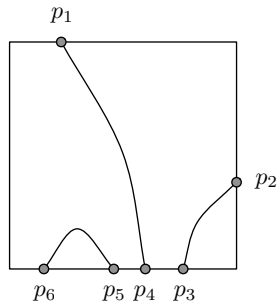
 choose (p, q) ;

$\mathcal{B} \leftarrow \mathcal{b} = [p, q]$;

$\mathcal{L} = \mathcal{L} \setminus p$;

$\mathcal{L} = \mathcal{L} \setminus q$;

end



Topology: non singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$\text{index}(\mathcal{C} \cap \partial\mathcal{D})$;

$\mathcal{L} \ll \text{order}(\mathcal{C} \cap \partial\mathcal{D})$;

while $\mathcal{L} \neq \emptyset$ **do**

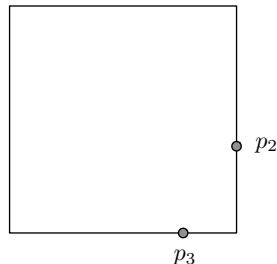
 choose (p, q) ;

$\mathcal{B} \leftarrow \mathfrak{b} = [p, q]$;

$\mathcal{L} = \mathcal{L} \setminus p$;

$\mathcal{L} = \mathcal{L} \setminus q$;

end



Topology: non singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$\text{index}(\mathcal{C} \cap \partial\mathcal{D})$;

$\mathcal{L} \ll \text{order}(\mathcal{C} \cap \partial\mathcal{D})$;

while $\mathcal{L} \neq \emptyset$ **do**

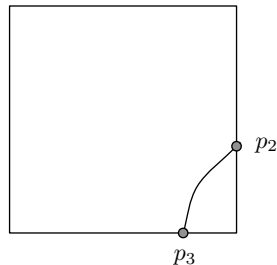
 choose (p, q) ;

$\mathcal{B} \leftarrow \mathfrak{b} = [p, q]$;

$\mathcal{L} = \mathcal{L} \setminus p$;

$\mathcal{L} = \mathcal{L} \setminus q$;

end



Topology: non singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$\text{index}(\mathcal{C} \cap \partial\mathcal{D})$;

$\mathcal{L} \ll \text{order}(\mathcal{C} \cap \partial\mathcal{D})$;

while $\mathcal{L} \neq \emptyset$ **do**

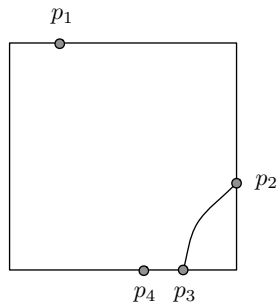
 choose (p, q) ;

$\mathcal{B} \leftarrow \mathfrak{b} = [p, q]$;

$\mathcal{L} = \mathcal{L} \setminus p$;

$\mathcal{L} = \mathcal{L} \setminus q$;

end



Topology: non singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$\text{index}(\mathcal{C} \cap \partial\mathcal{D})$;

$\mathcal{L} \ll \text{order}(\mathcal{C} \cap \partial\mathcal{D})$;

while $\mathcal{L} \neq \emptyset$ **do**

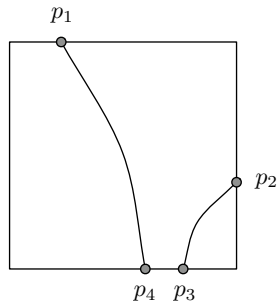
 choose (p, q) ;

$\mathcal{B} \leftarrow \mathfrak{b} = [p, q]$;

$\mathcal{L} = \mathcal{L} \setminus p$;

$\mathcal{L} = \mathcal{L} \setminus q$;

end



Topology: non singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$\text{index}(\mathcal{C} \cap \partial\mathcal{D})$;

$\mathcal{L} \ll \text{order}(\mathcal{C} \cap \partial\mathcal{D})$;

while $\mathcal{L} \neq \emptyset$ **do**

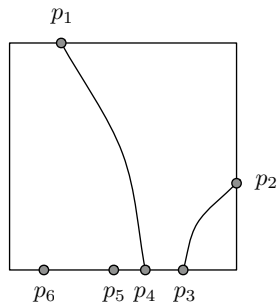
 choose (p, q) ;

$\mathcal{B} \leftarrow \mathcal{b} = [p, q]$;

$\mathcal{L} = \mathcal{L} \setminus p$;

$\mathcal{L} = \mathcal{L} \setminus q$;

end



Topology: non singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$\text{index}(\mathcal{C} \cap \partial\mathcal{D})$;

$\mathcal{L} \ll \text{order}(\mathcal{C} \cap \partial\mathcal{D})$;

while $\mathcal{L} \neq \emptyset$ **do**

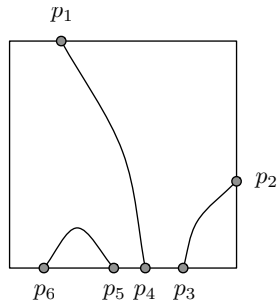
 choose (p, q) ;

$\mathcal{B} \leftarrow \mathfrak{b} = [p, q]$;

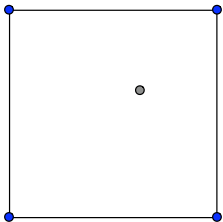
$\mathcal{L} = \mathcal{L} \setminus p$;

$\mathcal{L} = \mathcal{L} \setminus q$;

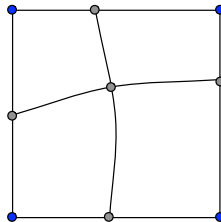
end



Regularity criteria: singular cells



(d) Isolated points



(e) Self-intersection

Regularity criteria: singular cells

Definition

A domain \mathcal{D} is *simply singular* for \mathcal{C} if $\mathcal{S} \cap \mathcal{D} = \{p\}$ and if the number n of half branches of \mathcal{C} at the singular point p is equal to $\#(\partial\mathcal{D} \cap \mathcal{C})$, the number of points of \mathcal{C} on the boundary of \mathcal{D} .

Proposition

Let \mathcal{D} be a simply singular domain. The topology of \mathcal{D} is conic, *i.e.* for any point p in the inside \mathcal{D} , $\mathcal{Z}(f) \cap \mathcal{D}$ can be deformed into $p \star (\partial\mathcal{D} \cap \mathcal{C})$.

How to count the number of branches ?

Regularity criteria: singular cells

Definition

A domain \mathcal{D} is *simply singular* for \mathcal{C} if $\mathcal{S} \cap \mathcal{D} = \{p\}$ and if the number n of half branches of \mathcal{C} at the singular point p is equal to $\#(\partial\mathcal{D} \cap \mathcal{C})$, the number of points of \mathcal{C} on the boundary of \mathcal{D} .

Proposition

Let \mathcal{D} be a simply singular domain. The topology of \mathcal{D} is conic, *i.e.* for any point p in the inside \mathcal{D} , $\mathcal{Z}(f) \cap \mathcal{D}$ can be deformed into $p \star (\partial\mathcal{D} \cap \mathcal{C})$.

How to count the number of branches ?

Regularity criteria: singular cells

Definition (Topological degree)

The *topological degree of F at p relative to \mathcal{D}* , denoted by $\deg[F, \mathcal{D}, p]$, is defined by

$$\deg[F, \mathcal{D}, p] = \sum_{\mathbf{x} \in \mathcal{D}: F(\mathbf{x})=p} \text{sign}(\det(J_F(\mathbf{x})))$$

Theorem (Khimshiashvili)

Suppose that p is the only root of $\nabla f = 0$ in \mathcal{D} . Then the number N of real half branches at p of the curve defined by $f(x, y) = f(p)$ is

$$N = 2(1 - \deg[\nabla f, \mathcal{D}, p])$$

Topology: singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

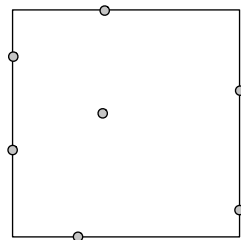
Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$Q = \{\mathcal{C} \cap \partial\mathcal{D}\}$;

forall $q_i \in Q$ **do**

$\mathcal{B} \leftarrow [s, q_i]$;

end



Topology: singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

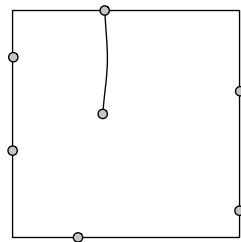
Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$Q = \{\mathcal{C} \cap \partial\mathcal{D}\}$;

forall $q_i \in Q$ **do**

$\mathcal{B} \leftarrow [s, q_i]$;

end



Topology: singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

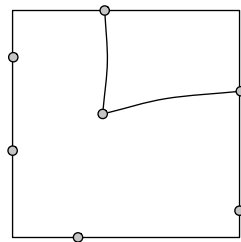
Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$Q = \{\mathcal{C} \cap \partial\mathcal{D}\}$;

forall $q_i \in Q$ **do**

$\mathcal{B} \leftarrow [s, q_i]$;

end



Topology: singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

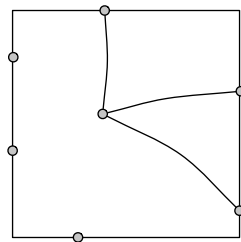
Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$Q = \{\mathcal{C} \cap \partial\mathcal{D}\}$;

forall $q_i \in Q$ **do**

$\mathcal{B} \leftarrow [s, q_i]$;

end



Topology: singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

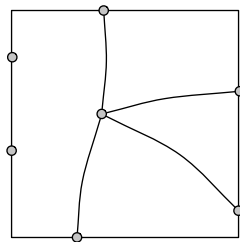
Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$Q = \{\mathcal{C} \cap \partial\mathcal{D}\}$;

forall $q_i \in Q$ **do**

$\mathcal{B} \leftarrow [s, q_i]$;

end



Topology: singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

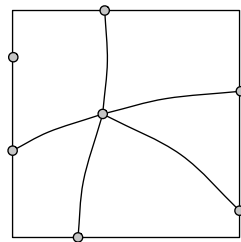
Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$Q = \{\mathcal{C} \cap \partial\mathcal{D}\}$;

forall $q_i \in Q$ **do**

$\mathcal{B} \leftarrow [s, q_i]$;

end



Topology: singular cells

Input: an algebraic curve \mathcal{C} and
a domain \mathcal{D}

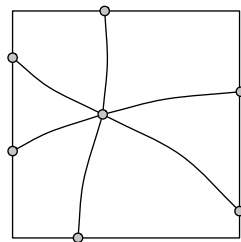
Output: the set \mathcal{B} of branches of
 \mathcal{C} in \mathcal{D}

$Q = \{\mathcal{C} \cap \partial\mathcal{D}\}$;

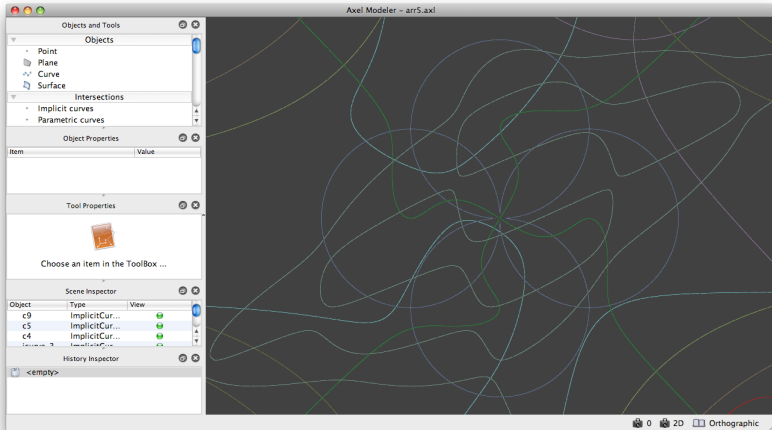
forall $q_i \in Q$ **do**

$\mathcal{B} \leftarrow [s, q_i]$;

end



Demonstration



Outline

- ① Generic algorithm
 - Computing regions
 - Locating conflicts
 - Updating regions
- ② Specialization for curves
 - Implicit curves
 - Parametric curves**
- ③ Specialization for surfaces
 - Implicit surfaces
 - Parametric surfaces
- ④ An algebraic geometric modeler
 - User perspective
 - Developer perspective

Definitions

Polynomial rational curve

A uniform rational polynomial curve is defined by the formula

$$c(t) = \begin{cases} x(t) \\ y(t) \end{cases}$$

where $x : \mathbb{R}[t] \mapsto \mathbb{R}$ and $y : \mathbb{R}[t] \mapsto \mathbb{R}$ are differentiable functions evaluated to obtain the image of $t \in \mathbb{R}[t]$ by c in \mathbb{R}^2 as the point $p_t = (x(t), y(t))$.

Definitions

B-Spline curve

A B-Spline curve is defined by the formula

$$c(t) = \sum_{i=0}^n \mathbf{p}_i B_{i,k,t}(t)$$

The complete representation of a B-spline curve consists of:

n The number of vertices

k The order

\mathbf{t} The knot vector: $\mathbf{t} = (t_1, t_2, \dots, t_{n+k})$

\mathbf{p} The control points: $p_{d,i}, d = 1 \dots dim, i = 1 \dots n$

Definitions

Critical points

Solve with univariate subdivision solver:

- $x'(t) = 0$
- $y'(t) = 0$

Self-intersection points

Solve with multivariate subdivision solver:

$$\begin{cases} x(t) - x(s) = 0 \\ y(t) - y(s) = 0 \end{cases}$$

Definitions

Curve-Curve intersection points

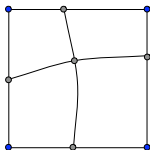
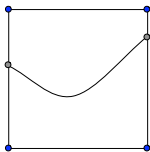
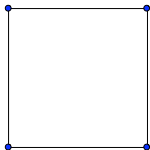
Solve with multivariate subdivision solver:

$$\begin{cases} x_k(s) = x_l(t) \\ y_k(s) = y_l(t) \end{cases}$$

Curve-Cell intersection points

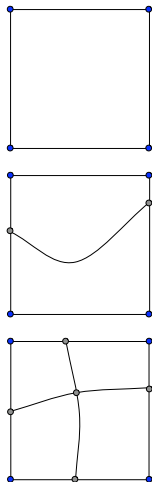
- $y(t) - y_{min} = 0, x_{min} \leq x(t) \leq x_{max}$
- $y(t) - y_{max} = 0, x_{min} \leq x(t) \leq x_{max}$
- $x(t) - x_{min} = 0, y_{min} \leq y(t) \leq y_{max}$
- $x(t) - x_{max} = 0, y_{min} \leq y(t) \leq y_{max}$

Regularity criteria and topology computation



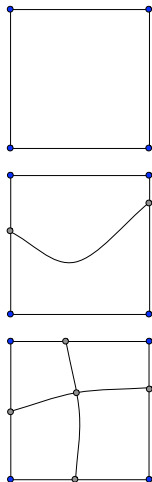
- Check that $\mathcal{C} \cap \partial\mathcal{D} = \emptyset$
- Check that there is at most one critical point.
- Check that $\#(\mathcal{C} \cap \partial\mathcal{D}) = 2$.
- Check that there is at most one self-intersection point s .
- Check that $\#(\mathcal{C} \cap \partial\mathcal{D}) = \star(s)$.

Regularity criteria and topology computation



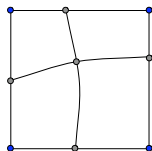
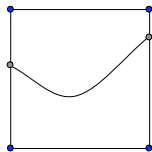
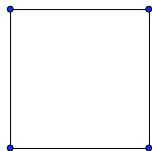
- Check that $\mathcal{C} \cap \partial\mathcal{D} = \emptyset$
- Check that there is at most one critical point.
- Check that $\#(\mathcal{C} \cap \partial\mathcal{D}) = 2$.
- Check that there is at most one self-intersection point s .
- Check that $\#(\mathcal{C} \cap \partial\mathcal{D}) = \star(s)$.

Regularity criteria and topology computation



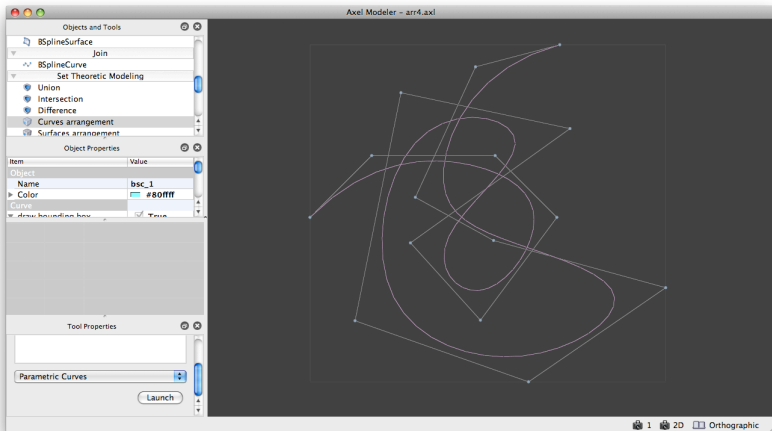
- Check that $\mathcal{C} \cap \partial\mathcal{D} = \emptyset$
- Check that there is at most one critical point.
- Check that $\#(\mathcal{C} \cap \partial\mathcal{D}) = 2$.
- Check that there is at most one self-intersection point s .
- Check that $\#(\mathcal{C} \cap \partial\mathcal{D}) = \star(s)$.

Regularity criteria and topology computation



- Check that $\mathcal{C} \cap \partial\mathcal{D} = \emptyset$
- Check that there is at most one critical point.
- Check that $\#(\mathcal{C} \cap \partial\mathcal{D}) = 2$.
- Check that there is at most one self-intersection point s .
- Check that $\#(\mathcal{C} \cap \partial\mathcal{D}) = \star(s)$.

Demonstration



Outline

- ① Generic algorithm
 - Computing regions
 - Locating conflicts
 - Updating regions
- ② Specialization for curves
 - Implicit curves
 - Parametric curves
- ③ **Specialization for surfaces**
 - Implicit surfaces
 - Parametric surfaces
- ④ An algebraic geometric modeler
 - User perspective
 - Developer perspective

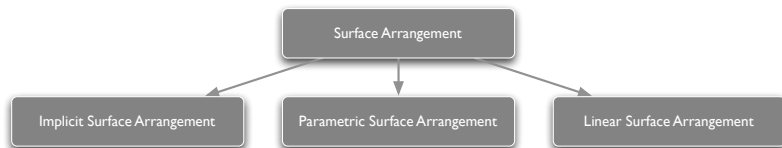
Specialization

Generic algorithm

- Global problem
- Any representation

Specific algorithm

- Local problem
- One representation



Specialization

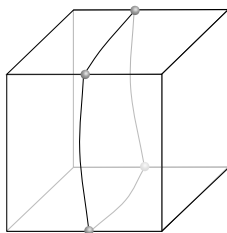
Characterize the situation:

- Near a 2-dimensional stratum, the topology is the same as a hyperplane
- Near a 1-dimensional stratum, the topology is the same as a cylinder
- Near a 0-dimensional stratum, the topology is the same as a cone

Specialization

Characterize the situation:

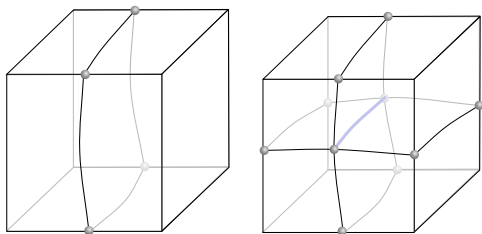
- Near a 2-dimensional stratum, the topology is the same as a hyperplane
- Near a 1-dimensional stratum, the topology is the same as a cylinder
- Near a 0-dimensional stratum, the topology is the same as a cone



Specialization

Characterize the situation:

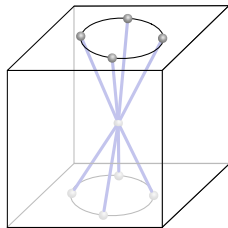
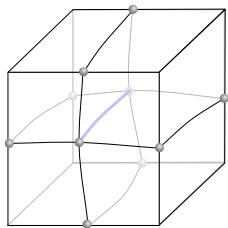
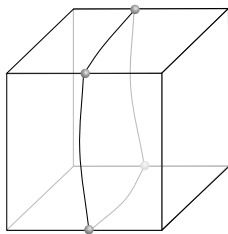
- Near a 2-dimensional stratum, the topology is the same as a hyperplane
- Near a 1-dimensional stratum, the topology is the same as a cylinder
- Near a 0-dimensional stratum, the topology is the same as a cone



Specialization

Characterize the situation:

- Near a 2-dimensional stratum, the topology is the same as a hyperplane
- Near a 1-dimensional stratum, the topology is the same as a cylinder
- Near a 0-dimensional stratum, the topology is the same as a cone



Outline

- ① Generic algorithm
 - Computing regions
 - Locating conflicts
 - Updating regions
- ② Specialization for curves
 - Implicit curves
 - Parametric curves
- ③ **Specialization for surfaces**
 - Implicit surfaces**
 - Parametric surfaces
- ④ An algebraic geometric modeler
 - User perspective
 - Developer perspective

Definitions

Monomial basis

We denote by $f_k(x, y, z) \in \mathbb{R}[x, y, z]$ the polynomial defining the implicit surface corresponding to the object o_k .

Bernstein basis

The specific operations will be performed on the Bernstein representation of f_k on $D = [a, b] \times [c, d] \times [e, f]$:

$$f_k(x, y, z) = \sum_{i=0}^{d_{x,k}} \sum_{j=0}^{d_{y,k}} \sum_{k=0}^{d_{z,k}} b_{i,j,k}^k B_{d_{x,k}}^i(x; a, b) B_{d_{y,k}}^j(y; c, d) B_{d_{z,k}}^k(z; e, f)$$

where $B_d^i(x; u, v) = \binom{d}{i} (x - u)^i (v - x)^{d-i} (v - u)^{-d}$.

Definitions

Critical points

Solve with multivariate subdivision solver:

$$\begin{cases} f_k(x, y, z) = 0 \\ \partial_y f_k(x, y, z) = 0 \\ \partial_z f_k(x, y, z) = 0 \end{cases} \quad \begin{cases} f_k(x, y, z) = 0 \\ \partial_x f_k(x, y, z) = 0 \\ \partial_z f_k(x, y, z) = 0 \end{cases} \quad \begin{cases} f_k(x, y, z) = 0 \\ \partial_x f_k(x, y, z) = 0 \\ \partial_y f_k(x, y, z) = 0 \end{cases}$$

Singular points

Solve with multivariate subdivision solver:

$$\begin{cases} f_k(x, y, z) = 0 \\ \partial_x f_k(x, y, z) = 0 \\ \partial_y f_k(x, y, z) = 0 \\ \partial_z f_k(x, y, z) = 0 \end{cases}$$

Definitions

Surface-Cell intersection curves

Variable substitution:

- $f(x, y_{min}, z)$ and $f(x, y_{max}, z)$ in x, z .
- $f(x_{min}, y, z)$ and $f(x_{max}, y, z)$ in y, z .
- $f(x, y, z_{min})$ and $f(x, y, z_{max})$ in x, y .

Surface-Surface intersection curves

The spatial implicit curve:

$$\begin{cases} f_k(x, y, z) = 0 \\ f_l(x, y, z) = 0 \end{cases}$$

Definitions

Definition (Polar variety)

The polar variety \mathcal{P}_f of a surface \mathcal{S} defined by the polynomial equation $f(x, y, z) = 0$ cuts the surface at the points of self-intersection and the points that have vertical tangents:

$$\begin{cases} f(x, y, z) = 0 \\ \partial_z f(x, y, z) = 0 \end{cases}$$

Spatial implicit curves

Let \mathcal{C} be a curve of \mathbb{R}^3 defined by $f(x, y, z)$ and $g(x, y, z)$.

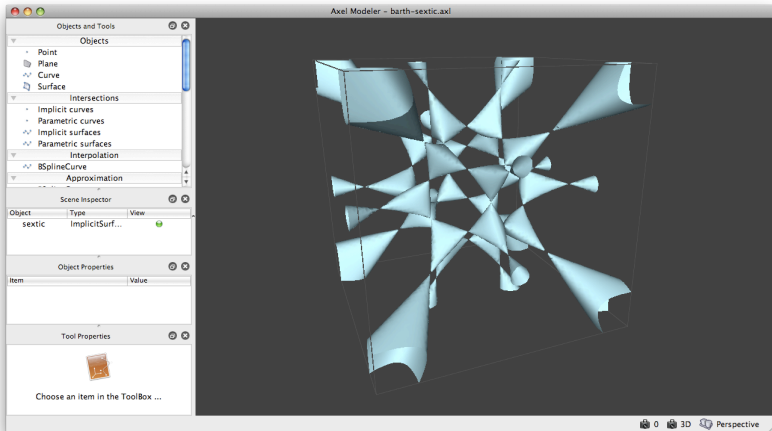
Examine the tangent vector field:

$$\mathbf{t} = \nabla(f) \wedge \nabla(g) = \begin{vmatrix} \mathbf{e}_x & \mathbf{e}_y & \mathbf{e}_z \\ \partial_x f & \partial_y f & \partial_z f \\ \partial_x g & \partial_y g & \partial_z g \end{vmatrix}$$

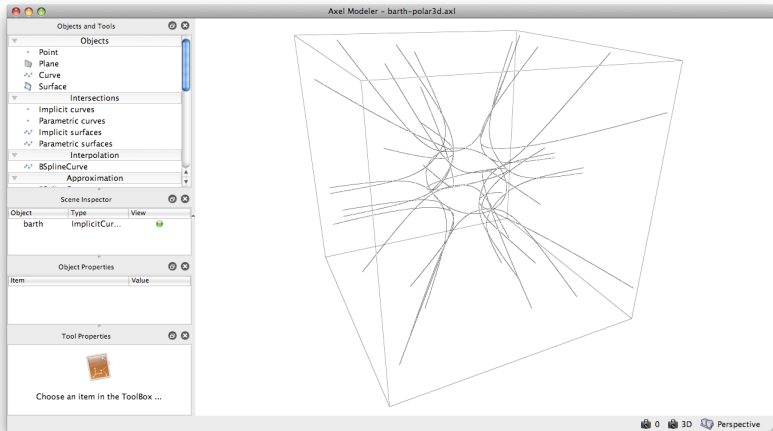
Remark

Singularities on the curve can be easily deduced, as \mathbf{t} vanishes at those points.

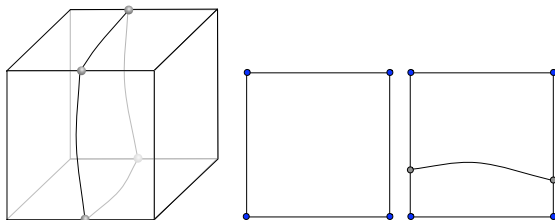
Spatial implicit curves



Spatial implicit curves

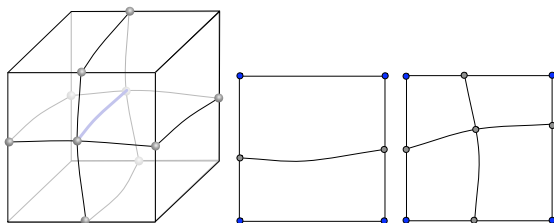


Regularity: 2-dimensional stratum



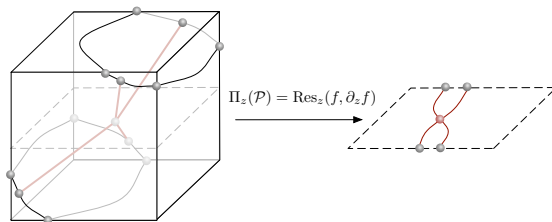
- Check each facet for 2-dimensional regularity:
 - empty cell
 - regular cell with one curve segment
- Check that the polar variety \mathcal{P} does not intersect \mathcal{D} ($\mathcal{P} \cap \mathcal{D} = \emptyset$).

Regularity: 1-dimensional stratum



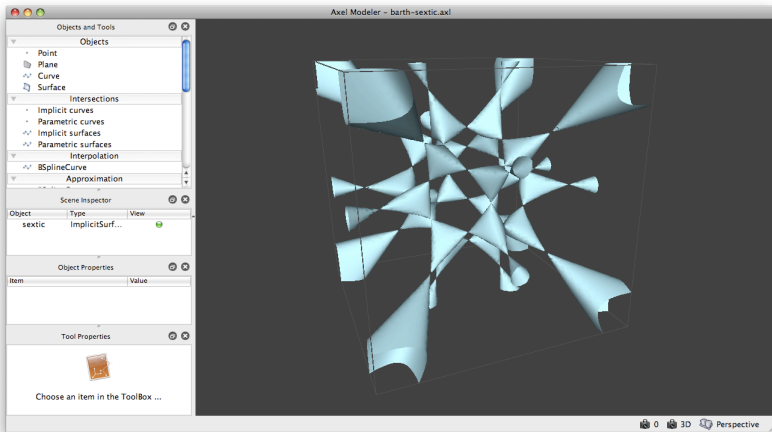
- Check each facet for 2-dimensional regularity:
 - regular cell with one curve segment
 - star shaped singularity with two curve segments
- Check that the polar variety \mathcal{P} features only one regular branch in \mathcal{D} ($\mathcal{P} \cap \mathcal{D} \neq \emptyset$).

Regularity: 0-dimensional stratum



- Check each facet for 2-dimensional regularity:
 - empty cell
 - regular cell with one curve segment
- Check that the polar variety features only one singular point in \mathcal{D} .
- Compute the projection $\Pi_z(\mathcal{P}) = \text{Res}_z(f, \partial_z f)$
- Check that $\Pi_z(\mathcal{P})$ is 2-regular

Topology



Outline

- ① Generic algorithm
 - Computing regions
 - Locating conflicts
 - Updating regions
- ② Specialization for curves
 - Implicit curves
 - Parametric curves
- ③ **Specialization for surfaces**
 - Implicit surfaces
 - Parametric surfaces
- ④ An algebraic geometric modeler
 - User perspective
 - Developer perspective

Definitions

Polynomial rational surface

A uniform rational polynomial surface is defined by the formula

$$s(u, v) = \begin{cases} x(u, v) \\ y(u, v) \\ z(u, v) \end{cases}$$

where $x : \mathbb{R}[u, v] \mapsto \mathbb{R}$, $y : \mathbb{R}[u, v] \mapsto \mathbb{R}$ and $z : \mathbb{R}[u, v] \mapsto \mathbb{R}$ are differentiable functions evaluated to obtain the image of $(u, v) \in \mathbb{R}[u, v]$ by s in \mathbb{R}^3 as the point $p_t = (x(u, v), y(u, v), z(u, v))$.

Definitions

B-Spline surface

A B-Spline surface is defined by the formula

$$s(u, v) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \mathbf{p}_{i,j} B_{i,k_1,u}(u) B_{j,k_2,v}(v)$$

The complete representation of a B-spline surface consists of:

- d* The dimension of the underlying Euclidean space
- n1* The number of vertices with respect to the first parameter
- n2* The number of vertices with respect to the second parameter
- k1* The order of the B-spline surface in the first parameter
- k2* The order of the B-spline surface in the second parameter
- u** The knot vector of the B-spline surface wrt. the first parameter
- v** The knot vector of the B-spline surface wrt. the second parameter
- p** The control points of the B-spline surface

Definitions

Surface - Cell edges intersection points

$$v_{x_i, y_j} = \{(x(s, t), y(s, t), z(s, t)) \in \mathcal{D} \mid (s, t) \in \mathcal{D}_p, x(u, v) = x_i, y(u, v) = y_i\}$$

$$v_{x_i, z_j} = \{(x(s, t), y(s, t), z(s, t)) \in \mathcal{D} \mid (s, t) \in \mathcal{D}_p, x(u, v) = x_i, z(u, v) = z_i\}$$

$$v_{y_i, z_j} = \{(x(s, t), y(s, t), z(s, t)) \in \mathcal{D} \mid (s, t) \in \mathcal{D}_p, y(u, v) = y_i, z(u, v) = z_i\}$$

where $x_i \in \{x_{min}, x_{max}\}$, $y_i \in \{y_{min}, y_{max}\}$ and $z_i \in \{z_{min}, z_{max}\}$.

Surface - Cell faces intersection curves

$$e_{x_i} = \{(y(s, t), z(s, t)) \mid (s, t) \in \mathcal{D}_p, x(s, t) = x_i\}$$

$$e_{y_i} = \{(x(s, t), z(s, t)) \mid (s, t) \in \mathcal{D}_p, y(s, t) = y_i\}$$

$$e_{z_i} = \{(x(s, t), y(s, t)) \mid (s, t) \in \mathcal{D}_p, z(s, t) = z_i\}$$

where $x_i \in \{x_{min}, x_{max}\}$, $y_i \in \{y_{min}, y_{max}\}$ and $z_i \in \{z_{min}, z_{max}\}$.

Definitions

Tangent vectors

$$T_u(u, v) = \begin{pmatrix} \partial_u x(u, v) \\ \partial_u y(u, v) \\ \partial_u z(u, v) \end{pmatrix} \quad T_v(u, v) = \begin{pmatrix} \partial_v x(u, v) \\ \partial_v y(u, v) \\ \partial_v z(u, v) \end{pmatrix}$$

Normal vector

$$N(u, v) = T_u(u, v) \wedge T_v(u, v)$$

Definitions

Critical points

$$\begin{cases} N_y(u, v) = 0 \\ N_z(u, v) = 0 \end{cases} \quad \begin{cases} N_x(u, v) = 0 \\ N_z(u, v) = 0 \end{cases} \quad \begin{cases} N_x(u, v) = 0 \\ N_y(u, v) = 0 \end{cases}$$

Singular points

$$\begin{cases} N_x(u, v) = 0 \\ N_y(u, v) = 0 \\ N_z(u, v) = 0 \end{cases}$$

Definitions

Intersection and Self-intersection

- Sampling methods
- Algebraic methods
 - Analytic method
 - Sweeping method
 - Subdivision method

Sampling methods Work on approximations

Algebraic methods Work on the representation

Analytic methods Heavy computational cost

Sweeping methods Numerical errors at events

Subdivision methods Filters computations

Definitions

Intersection and Self-intersection

- Sampling methods
- Algebraic methods
 - Analytic method
 - Sweeping method
 - Subdivision method

Sampling methods Work on approximations

Algebraic methods Work on the representation

Analytic methods Heavy computational cost

Sweeping methods Numerical errors at events

Subdivision methods Filters computations

Definitions

Intersection and Self-intersection

- Sampling methods
- Algebraic methods
 - Analytic method
 - Sweeping method
 - Subdivision method

Sampling methods Work on approximations

Algebraic methods Work on the representation

Analytic methods Heavy computational cost

Sweeping methods Numerical errors at events

Subdivision methods Filters computations

Definitions

Intersection and Self-intersection

- Sampling methods
- Algebraic methods
 - Analytic method
 - Sweeping method
 - Subdivision method

Sampling methods Work on approximations

Algebraic methods Work on the representation

Analytic methods Heavy computational cost

Sweeping methods Numerical errors at events

Subdivision methods Filters computations

Definitions

Intersection and Self-intersection

- Sampling methods
- Algebraic methods
 - Analytic method
 - Sweeping method
 - Subdivision method

Sampling methods Work on approximations

Algebraic methods Work on the representation

Analytic methods Heavy computational cost

Sweeping methods Numerical errors at events

Subdivision methods Filters computations

Definitions

Intersection and Self-intersection

- Sampling methods
- Algebraic methods
 - Analytic method
 - Sweeping method
 - Subdivision method

Sampling methods Work on approximations

Algebraic methods Work on the representation

Analytic methods Heavy computational cost

Sweeping methods Numerical errors at events

Subdivision methods Filters computations

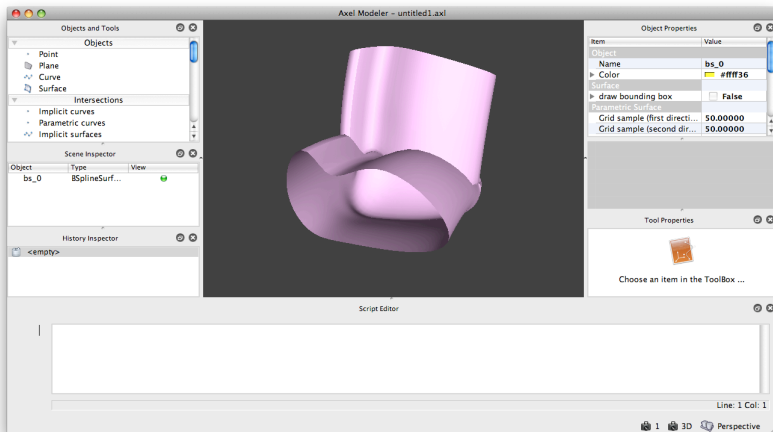
Outline

- ① Generic algorithm
 - Computing regions
 - Locating conflicts
 - Updating regions
- ② Specialization for curves
 - Implicit curves
 - Parametric curves
- ③ Specialization for surfaces
 - Implicit surfaces
 - Parametric surfaces
- ④ An algebraic geometric modeler
 - User perspective
 - Developer perspective

Generic algorithm
Specialization for curves
Specialization for surfaces
An algebraic geometric modeler

User perspective
Developer perspective

An algebraic geometric modeling environment



Outline

- ① Generic algorithm
 - Computing regions
 - Locating conflicts
 - Updating regions
- ② Specialization for curves
 - Implicit curves
 - Parametric curves
- ③ Specialization for surfaces
 - Implicit surfaces
 - Parametric surfaces
- ④ An algebraic geometric modeler
 - User perspective
 - Developer perspective

Objects

- Points
- Planes
- Curves
- Surfaces

Objects

- Points
- Planes
- Curves
- Surfaces

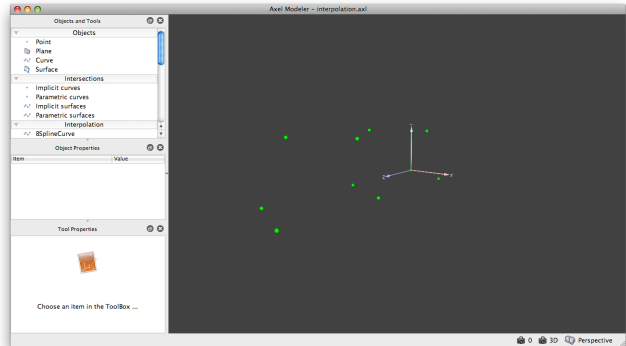


Figure: Points.

Objects

- Points
- Planes
- Curves
- Surfaces

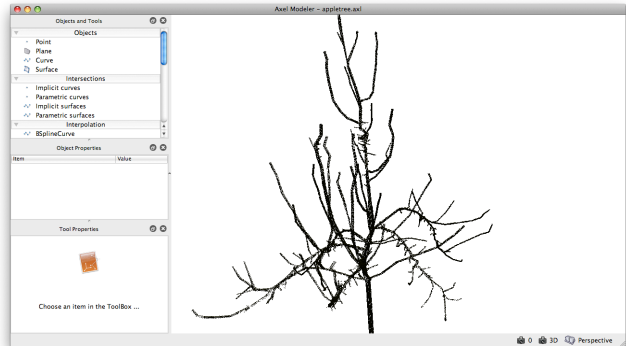


Figure: Pointsets.

Objects

- Points
- Planes
- Curves
- Surfaces

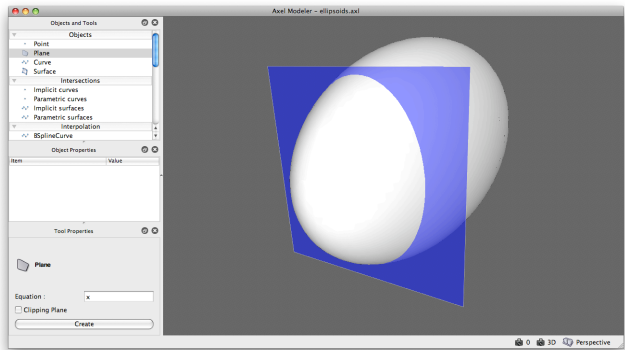


Figure: Planes.

Objects

- Points
- Planes
- Curves
- Surfaces

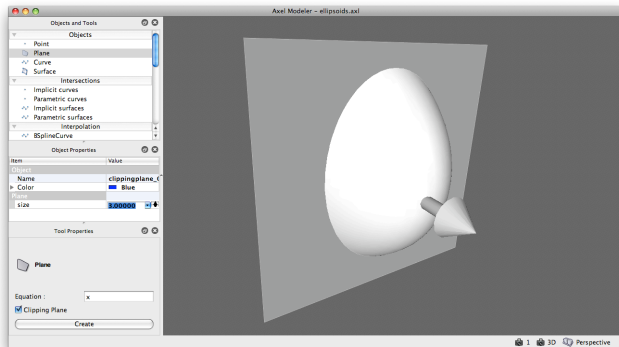


Figure: Clipping planes.

Objects

- Points
- Planes
- Curves
- Surfaces

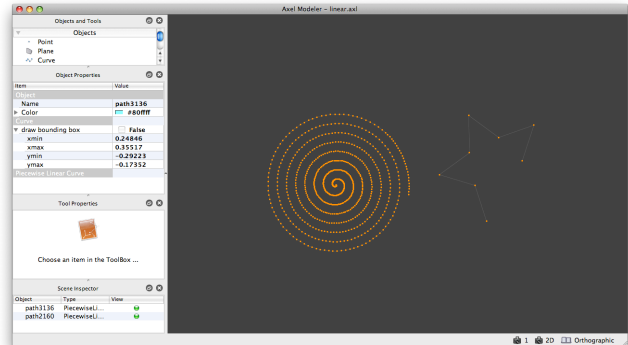


Figure: Piecewise linear curves.

Objects

- Points
- Planes
- Curves
- Surfaces

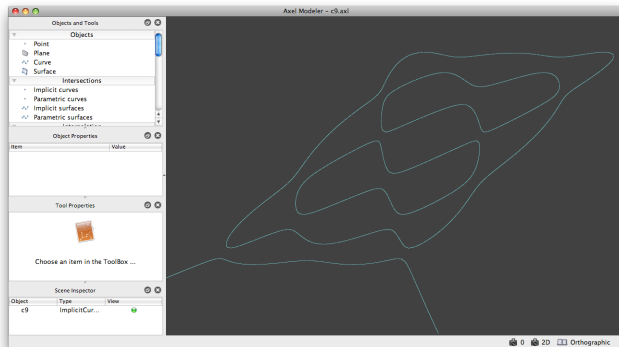


Figure: Implicit curves.

Objects

- Points
- Planes
- Curves
- Surfaces

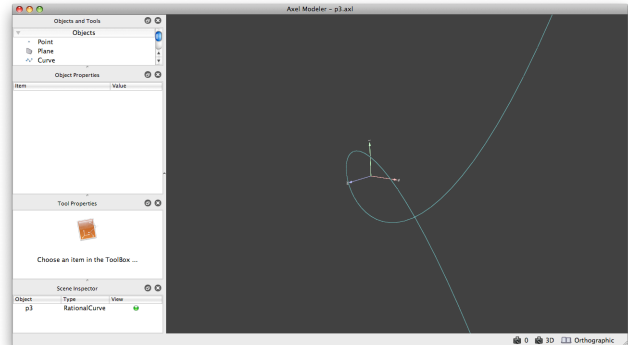


Figure: Rational curves.

Objects

- Points
- Planes
- Curves
- Surfaces

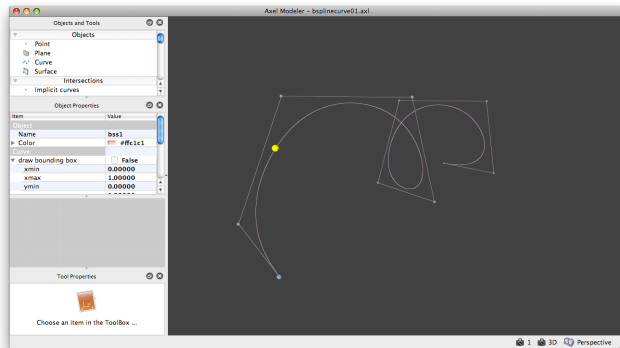


Figure: B-spline curves.

Objects

- Points
- Planes
- Curves
- Surfaces

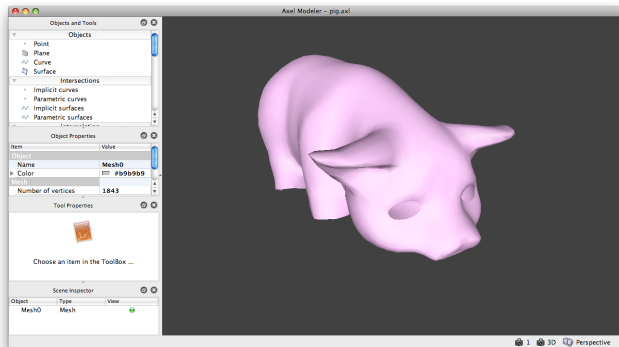


Figure: Piecewise linear surfaces.

Objects

- Points
- Planes
- Curves
- Surfaces

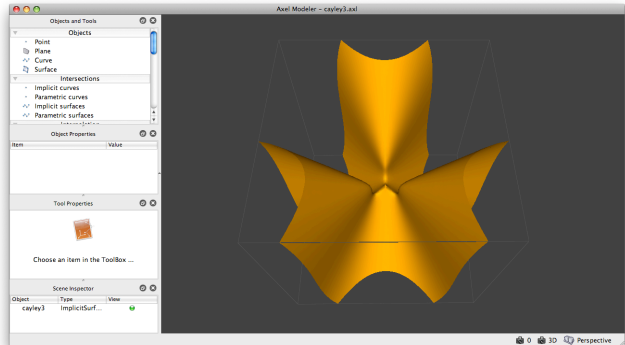


Figure: Implicit surfaces.

Objects

- Points
- Planes
- Curves
- Surfaces

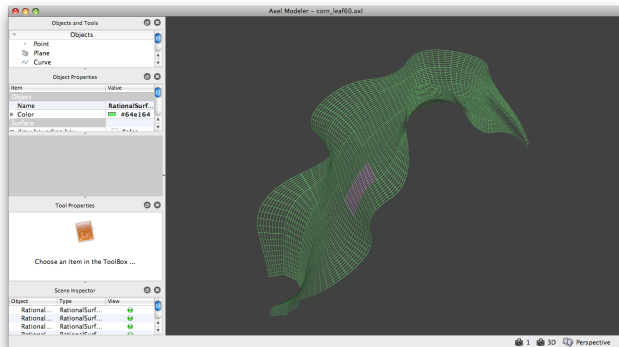


Figure: Rational surfaces.

Objects

- Points
- Planes
- Curves
- Surfaces

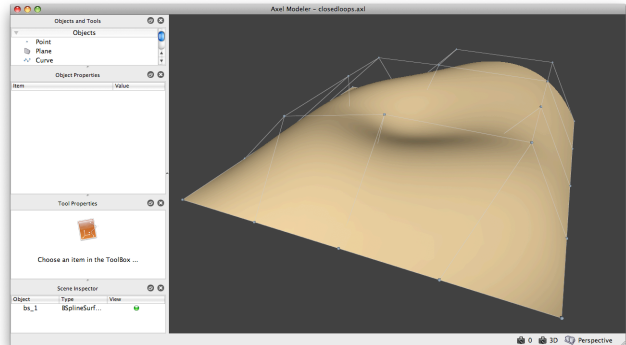
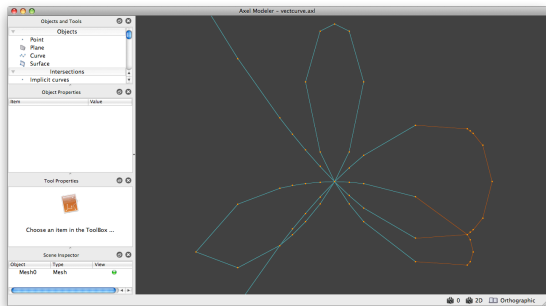


Figure: B-spline surfaces.

Tools

- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

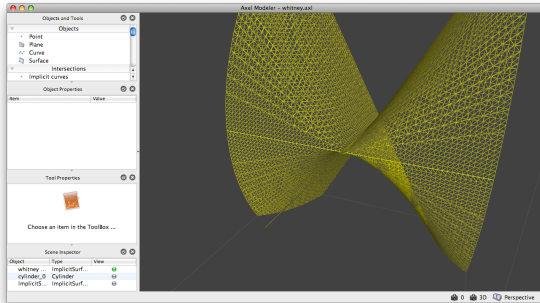
Tools



- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

Figure: Implicit curves topology.

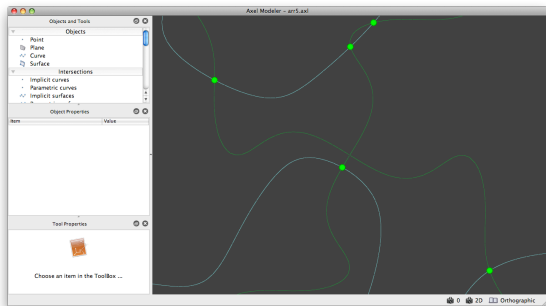
Tools



- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

Figure: Implicit surfaces topology.

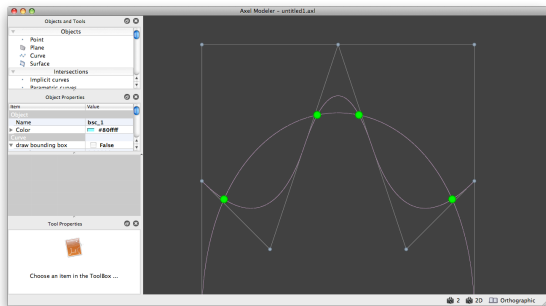
Tools



- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

Figure: Implicit curves intersection.

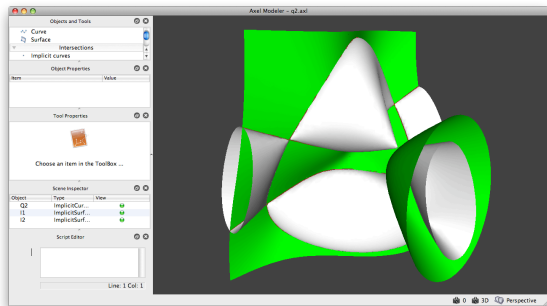
Tools



- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

Figure: Parametric curves intersection.

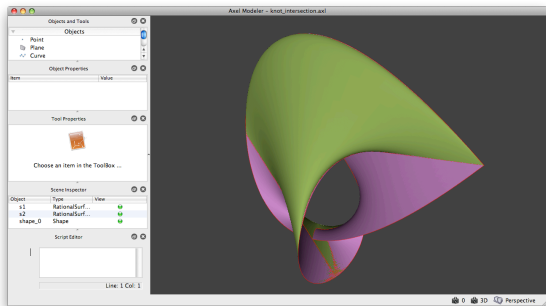
Tools



- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

Figure: Implicit surfaces intersection.

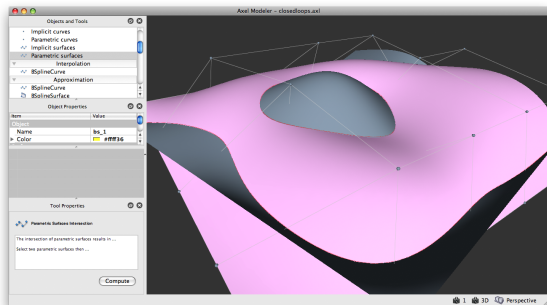
Tools



- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

Figure: Rational surfaces intersection.

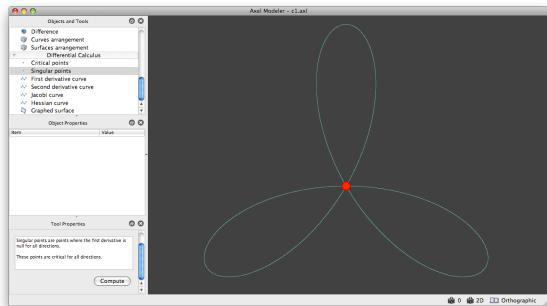
Tools



- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

Figure: B-spline surfaces intersection.

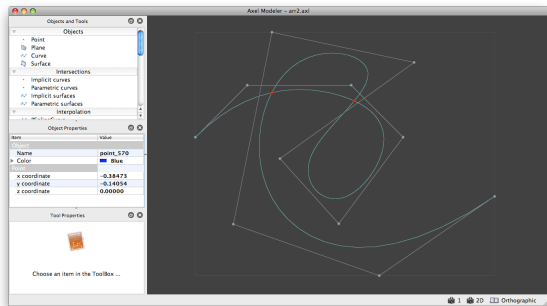
Tools



- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

Figure: Implicit curve self-intersection.

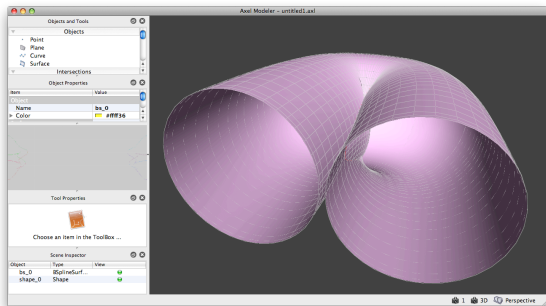
Tools



- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

Figure: Parametric curve self-intersection.

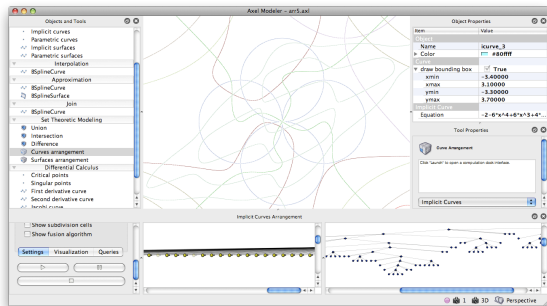
Tools



- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

Figure: Parametric surface self-intersection.

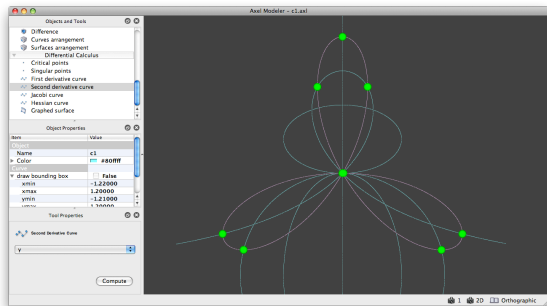
Tools



- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

Figure: Implicit curves arrangement.

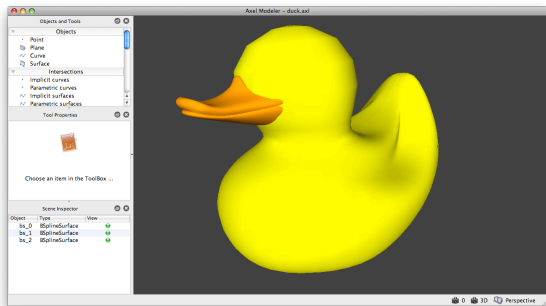
Tools



- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

Figure: Critical point and derivative curves.

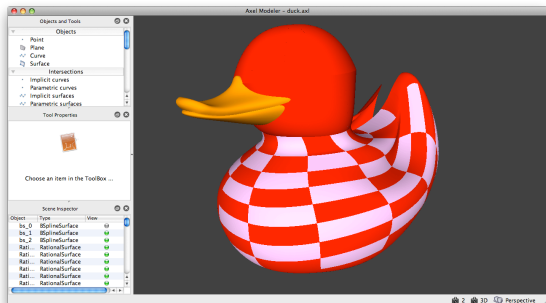
Tools



- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

Figure: Approximation by rational patches.

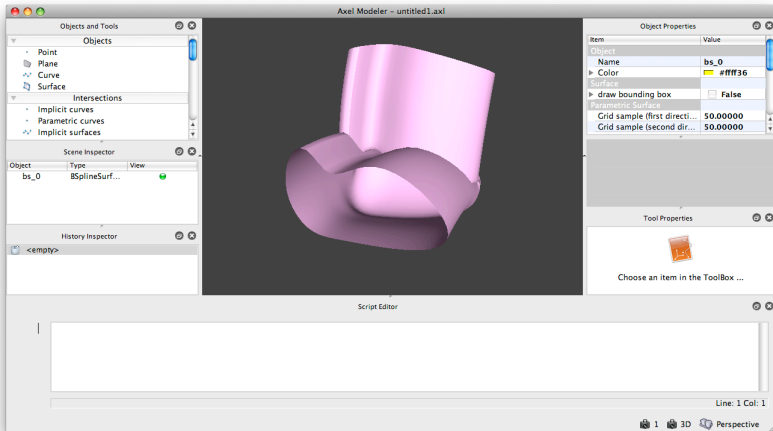
Tools



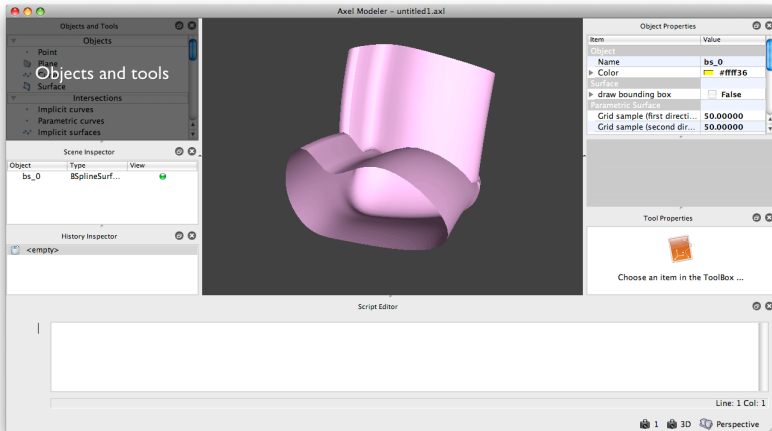
- Topology
- Intersection
- Self-intersection
- Arrangement
- Differential geometry
- Interpolation
- Approximation

Figure: Approximation by rational patches.

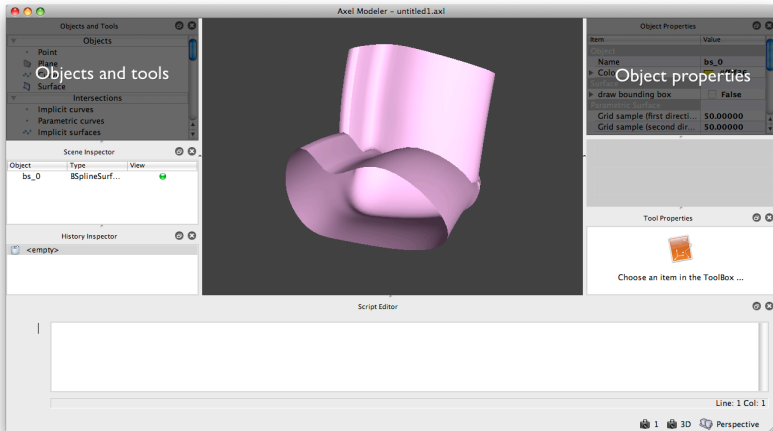
Graphical User Interface



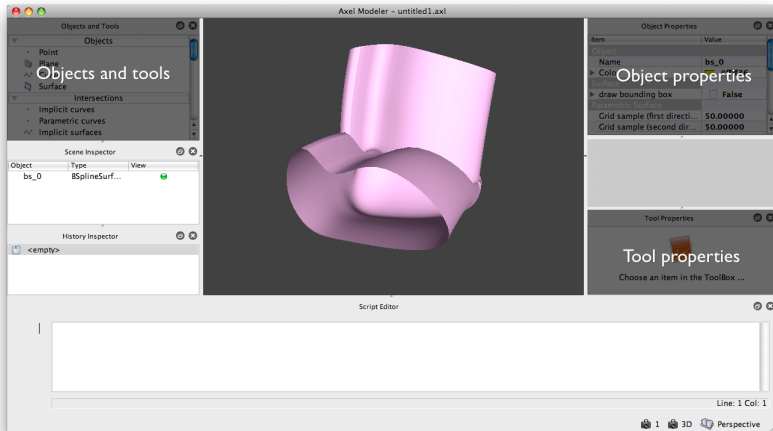
Graphical User Interface



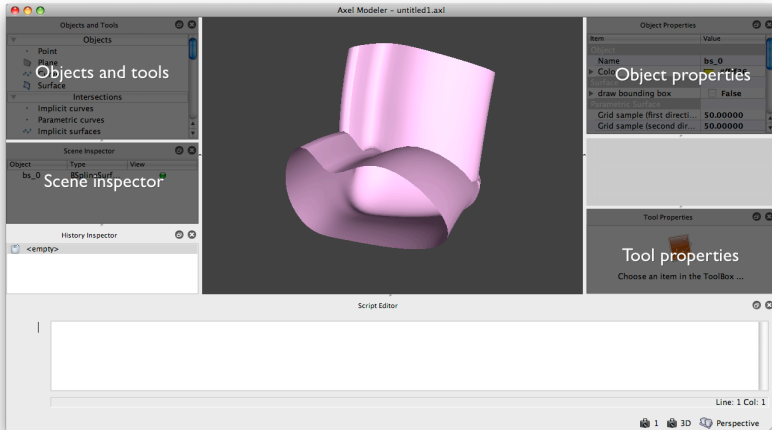
Graphical User Interface



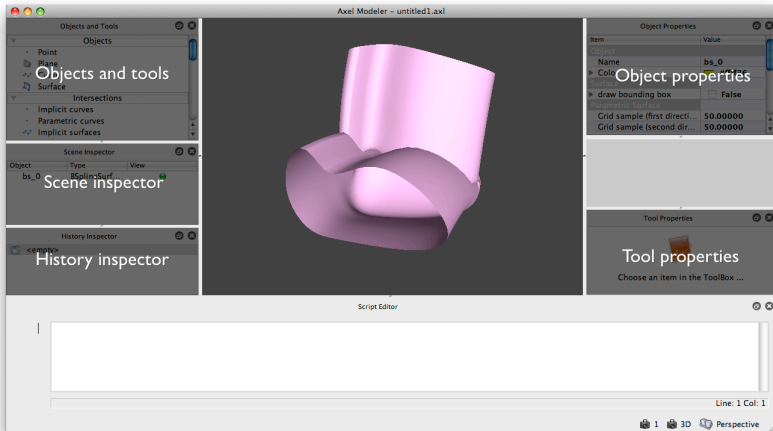
Graphical User Interface



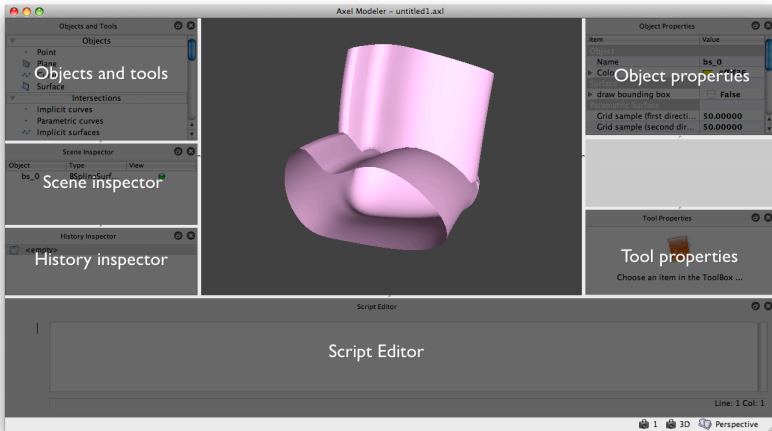
Graphical User Interface



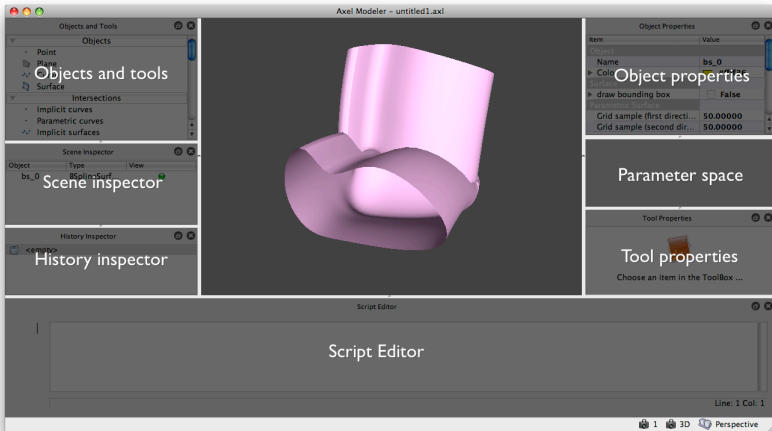
Graphical User Interface



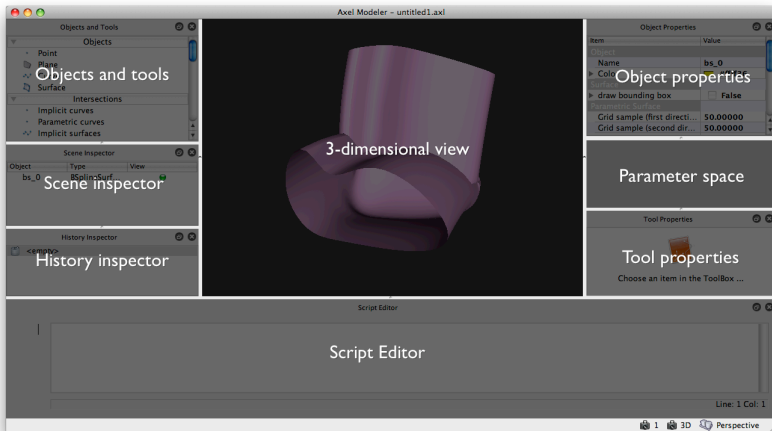
Graphical User Interface



Graphical User Interface



Graphical User Interface



File Interface

A B-spline curve.

```
<axl>
  <curve type="bspline" name="bsc1">
    <dimension>3</dimension>
    <number>10</number>
    <order>4</order>
    <knots>0 0 0 0 1 2 3 4 5 6 7 7 7 7</knots>
    <points>
      0 0 0 1 0 0.5 1 1 1 0 1 1.5 0 0 2
      1 0 2.5 1 1 3 0 1 3.5 0 0 4 1 0 4.5
    </points>
  </curve>
</axl>
```


Script Interface

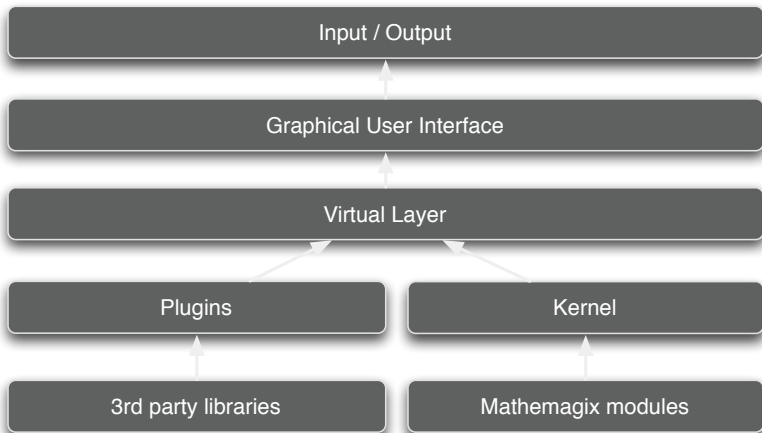
Creating a B-spline curve.

```
var white = new Color(255, 255, 255) ;  
Viewer.setBackgroundColor(white) ;  
Camera.setPosition(0.0, 1.0, 0.0) ;  
  
var p1 = new Point(0.0 0.0 0.0) ;  
var p2 = new Point(1.0 0.0 0.5) ;  
...  
var p8 = new Point(0.0 1.0 3.5) ;  
var p9 = new Point(0.0 0.0 4.0) ;  
var p10 = new Point(1.0 0.0 4.5) ;  
  
var curve = ToolManager.interpolate() ;  
  
ObjectManager.addObject(curve) ;
```

Outline

- ① Generic algorithm
 - Computing regions
 - Locating conflicts
 - Updating regions
- ② Specialization for curves
 - Implicit curves
 - Parametric curves
- ③ Specialization for surfaces
 - Implicit surfaces
 - Parametric surfaces
- ④ An algebraic geometric modeler
 - User perspective
 - Developer perspective

Framework



Virtual hierarchy of objects

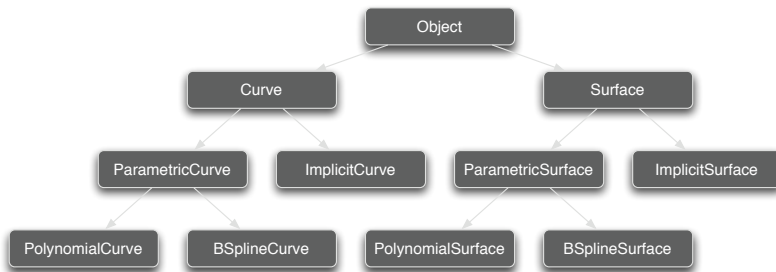


Figure: Virtual hierarchy of objects.

Virtual hierarchy of tools

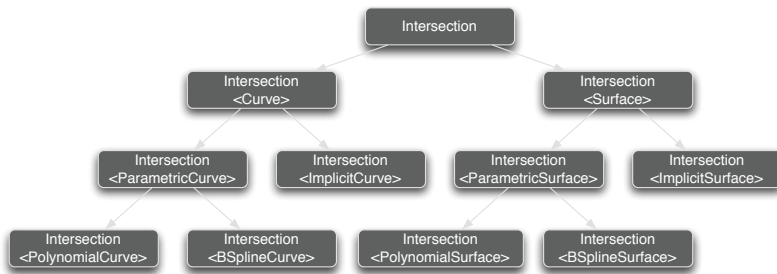


Figure: Virtual hierarchy of tools.

Kernel system

Algebraic Kernel

- Univariate polynomials
- Multivariate polynomials
- Univariate bernstein solvers
- Multivariate bernstein solvers
- Embeds Mathemagix

Geometric Kernel

- Topology
- Intersection
- Approximation
- Embeds Sisl

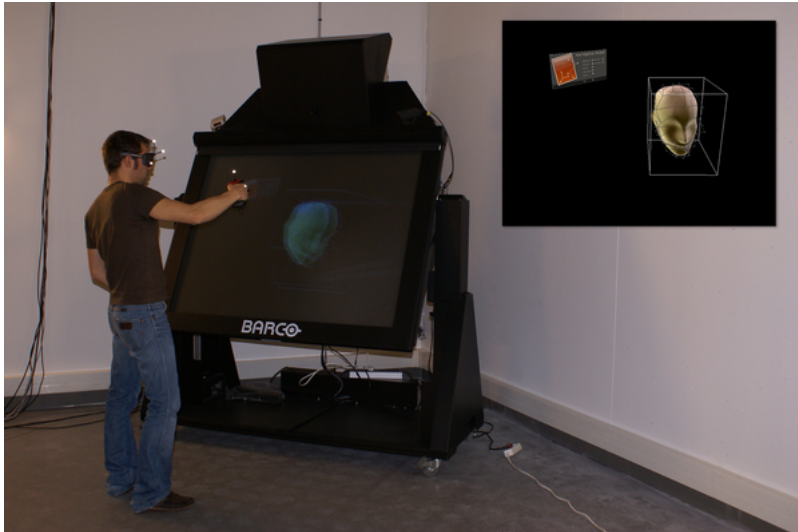
Plugin system

- Extend main application functionalities
- Define new objects
- Define new tools
- Glue external libraries
- Hardware connection

Generic algorithm
Specialization for curves
Specialization for surfaces
An algebraic geometric modeler

User perspective
Developer perspective

A virtual modeling environment



Summary

- Generic algorithm using a subdivision scheme
- Specialization for curves
 - Efficient topology
 - Efficient arrangement
- Specialization for surfaces
 - Hints for implicit surfaces
 - Hints for parametric surfaces
- Proof of concept implementation
 - Modular architecture of objects
 - Modular architecture of tools

Outlook

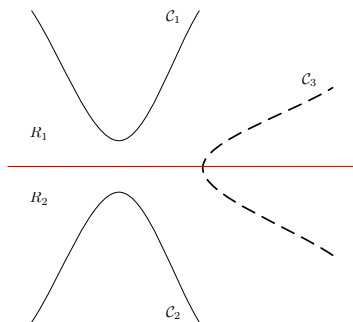
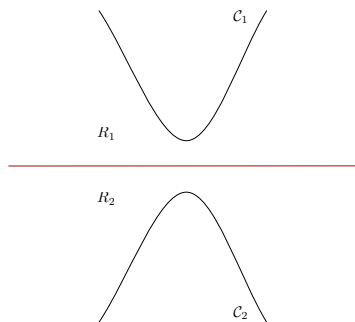
- Generic boolean operators
- Generic Voronoi diagrams
- Generic trimming procedures

Outlook

- Generic boolean operators
- Generic Voronoi diagrams
- Generic trimming procedures

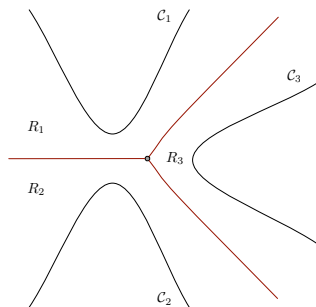
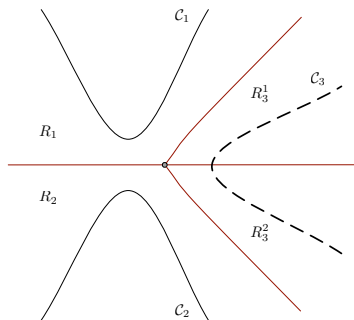
Outlook

- Generic boolean operators
- Generic Voronoi diagrams
- Generic trimming procedures



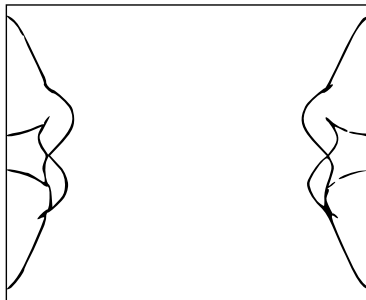
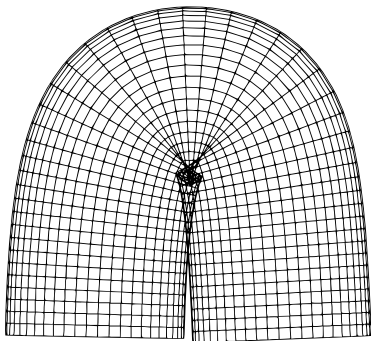
Outlook

- Generic boolean operators
- Generic Voronoi diagrams
- Generic trimming procedures



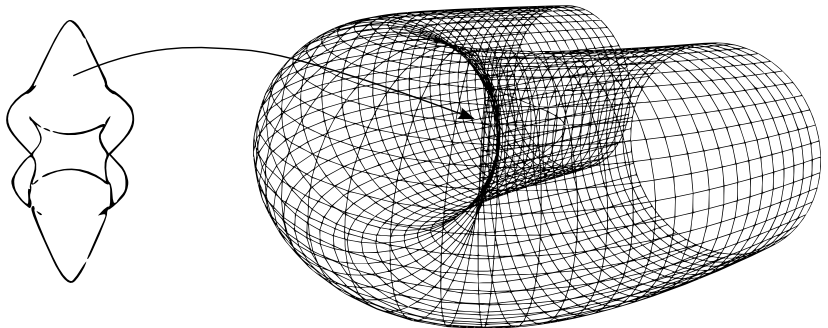
Outlook

- Generic boolean operators
- Generic Voronoi diagrams
- Generic trimming procedures



Outlook

- Generic boolean operators
- Generic Voronoi diagrams
- Generic trimming procedures



<http://axel.inria.fr>

