DCEL: Doubly-Connected Edge List

Claudio Mirolo

Dip. di Scienze Matematiche, Informatiche e Fisiche
Università di Udine, via delle Scienze 206 – Udine
claudio.mirolo@uniud.it

Computational Geometry
www.dimi.uniud.it/claudio
Outline

1. DCEL data structure
   - representation
   - example

2. Application of DCELs
   - overlay of two subdivisions
   - plane sweep: edges and vertices
   - plane sweep: faces

3. Further annotations
What about?

- Planar subdivision

- Planar embedding of a graph
  - node $\rightarrow$ vertex
  - arc $\rightarrow$ edge
  - + face
Planar subdivision

Planar embedding of a graph

- node $\rightarrow$ vertex
- arc $\rightarrow$ edge
- + face
Planar subdivision

Planar embedding of a graph

- node → vertex
- arc → edge
- + face
What about?

- Planar subdivision

- Planar embedding of a graph
  - node $\rightarrow$ vertex
  - arc $\rightarrow$ edge
  - + face
What about?

- Planar subdivision
- Planar embedding of a graph
  - node → vertex
  - arc → edge
  - + face
Outline

1. DCEL data structure
   - representation
   - example

2. Application of DCELs
   - overlay of two subdivisions
   - plane sweep: edges and vertices
   - plane sweep: faces

3. Further annotations
DCEL: Basic ingredients

Basic items:

- \textit{vertices}
- \textit{edges} = Line segments
- \textit{faces}

As well as topological relationships between such items: \textit{incidence}
DCEL: Basic ingredients

Basic items:

- vertices
- edges = Line segments
- faces

As well as topological relationships between such items: incidence
DCEL: Basic ingredients

Basic items:

- *vertices*
- *edges* = Line segments
- *faces*

As well as topological relationships between such items: *incidence*
Basic *local* access operations:

- walk around the boundary of a given face
- move from a face to an adjacent one (via a common edge)
- visit all the edges around a given vertex

Possibly, store/get additional data associated to, e.g., a face: *attribute information*
DCEL: Basic access operations

Basic *local* access operations:

- walk around the boundary of a given face
- move from a face to an adjacent one (via a common edge)
- visit all the edges around a given vertex

Possibly, store/get additional data associated to, e.g., a face: *attribute information*
Basic local access operations:

- walk around the boundary of a given face
- move from a face to an adjacent one (via a common edge)
- visit all the edges around a given vertex

Possibly, store/get additional data associated to, e.g., a face: attribute information
DCEL: Basic access operations

Basic *local* access operations:

- walk around the boundary of a given face
- move from a face to an adjacent one (via a common edge)
- visit all the edges around a given vertex

Possibly, store/get additional data associated to, e.g., a face: *attribute information*
DCEL: Representational tricks

One record for each item...

- edge $\rightarrow$ two directed *half-edges* ...
- $\ldots$ $\rightarrow$ *twin* half-edges
- face lies to the *left* of its bounding half-edges ...
DCEL: Representational tricks

One record for each item... 

- edge \rightarrow two directed half-edges...

- ... \rightarrow twin half-edges

- face lies to the left of its bounding half-edges...
DCEL: Representational tricks

One record for each item...

- edge → two directed half-edges...
- ... → twin half-edges
- face lies to the left of its bounding half-edges...
One record for each item...

- edge $\rightarrow$ two directed *half-edges* ...
- ... $\rightarrow$ *twin* half-edges
- face lies to the *left* of its bounding half-edges ...
DCEL: Representational tricks

... including its possible *holes*

Remark “walking” direction!
DCEL: Representational tricks

... including its possible holes

Remark “walking” direction!
Vertex record \( v \):

- pair of coordinates \((x, y)\)
- pointer to (any) incident half-edge, with \( v \) as its origin
DCEL: Vertices

Vertex record \( v \):

- pair of coordinates \((x, y)\)
- pointer to (any) incident half-edge, with \( v \) as its origin
DCEL: Vertices

Vertex record \( v \):

- pair of coordinates \((x, y)\)
- pointer to (any) incident half-edge, with \( v \) as its origin
DCEL: Vertices

Vertex record \( v \):

- pair of coordinates \( (x, y) \)
- pointer to (any) incident half-edge, with \( v \) as its origin
DCEL: Faces

Face record  \( f \):

- pointer to (any) half-edge of its outer boundary, with \( f \) lying to the left of this half-edge

- list of pointers to (any) half-edge of each inner boundary, with \( f \) still lying to the left of these half-edges
DCEL: Faces

Face record $f$:

- pointer to (any) half-edge of its outer boundary, with $f$ lying to the left of this half-edge
- list of pointers to (any) half-edge of each inner boundary, with $f$ still lying to the left of these half-edges
DCEL: Faces

Face record $f$:

- pointer to (any) half-edge of its outer boundary, with $f$ lying to the left of this half-edge
- list of pointers to (any) half-edge of each inner boundary, with $f$ still lying to the left of these half-edges
DCEL: Faces

Face record $f$:

- pointer to (any) half-edge of its outer boundary, with $f$ lying to the left of this half-edge
- list of pointers to (any) half-edge of each inner boundary, with $f$ still lying to the left of these half-edges
DCEL: Faces

Face record $f$

- pointer to (any) half-edge of its outer boundary, with $f$ lying to the left of this half-edge

- list of pointers to (any) half-edge of each inner boundary, with $f$ still lying to the left of these half-edges
**DCEL: Half-edges**

Half-edge record $e$:

- *source* vertex (origin of $e$)
- *incident* face, i.e. face to its left
- *twin* half-edge
- *next* half-edge, along boundary of incident face
- *previous* half-edge, along same boundary
DCEL: Half-edges

Half-edge record $e$:

- **source** vertex (origin of $e$)
- incident face, i.e. face to its left
- **twin** half-edge
- **next** half-edge, along boundary of incident face
- **previous** half-edge, along same boundary
Half-edge record  \( e \):

- \textit{source} vertex (origin of \( e \))
- incident face, i.e. face to its left
- \textit{twin} half-edge
- \textit{next} half-edge, along boundary of incident face
- \textit{previous} half-edge, along same boundary
DCEL: Half-edges

Half-edge record $e$:

- **source** vertex (origin of $e$)
- incident face, i.e. face to its left
- **twin** half-edge
- **next** half-edge, along boundary of incident face
- **previous** half-edge, along same boundary
DCEL: Half-edges

Half-edge record $e$:

- **source** vertex (origin of $e$)
- incident face, i.e. face to its left
- **twin** half-edge
- **next** half-edge, along boundary of incident face
- **previous** half-edge, along same boundary
DCEL: Half-edges

Half-edge record \( e \):

- **source** vertex (origin of \( e \))
- incident face, i.e. face to its left
- **twin** half-edge
- **next** half-edge, along boundary of incident face
- **previous** half-edge, along same boundary
DCEL: Half-edges

Half-edge record \( e \) :

- *source* vertex (origin of \( e \))
- incident face, i.e. face to its left
- *twin* half-edge
- *next* half-edge, along boundary of incident face
- *previous* half-edge, along same boundary
Example

DCEL data structure
Application of DCELs
Further annotations

Example

C. Mirolo
DCEL
Example

DCEL data structure
Application of DCELs
Further annotations

representation
example

C. Mirolo
DCEL
### Vertex records

<table>
<thead>
<tr>
<th></th>
<th>coord</th>
<th>edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td>(0.0, 2.5)</td>
<td>$e_{12}$</td>
</tr>
<tr>
<td>$v_2$</td>
<td>($-2.5$, 0.0)</td>
<td>$e_{23}$</td>
</tr>
<tr>
<td>$v_3$</td>
<td>(0.0, 0.0)</td>
<td>$e_{35}$</td>
</tr>
<tr>
<td>$v_4$</td>
<td>(2.5, 0.0)</td>
<td>$e_{41}$</td>
</tr>
<tr>
<td>$v_5$</td>
<td>(0.0, $-2.5$)</td>
<td>??</td>
</tr>
</tbody>
</table>

![DCEL diagram](image)
### Vertex records

<table>
<thead>
<tr>
<th>vertex</th>
<th>coord</th>
<th>edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td>(0.0, 2.5)</td>
<td>$e_{12}$</td>
</tr>
<tr>
<td>$v_2$</td>
<td>(-2.5, 0.0)</td>
<td>$e_{23}$</td>
</tr>
<tr>
<td>$v_3$</td>
<td>(0.0, 0.0)</td>
<td>$e_{35}$</td>
</tr>
<tr>
<td>$v_4$</td>
<td>(2.5, 0.0)</td>
<td>$e_{41}$</td>
</tr>
<tr>
<td>$v_5$</td>
<td>(0.0, -2.5)</td>
<td>$e_{53}$</td>
</tr>
</tbody>
</table>
## Face records

<table>
<thead>
<tr>
<th></th>
<th>outer</th>
<th>inner</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_U$</td>
<td>—</td>
<td>$&lt; e_{32} &gt;$</td>
</tr>
<tr>
<td>$f_A$</td>
<td>$e_{12}$</td>
<td>??</td>
</tr>
<tr>
<td>$f_B$</td>
<td>$e_{13}$</td>
<td>??</td>
</tr>
</tbody>
</table>
## Face records

<table>
<thead>
<tr>
<th></th>
<th>outer</th>
<th>inner</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_U$</td>
<td>—</td>
<td>$&lt; e_{32} &gt;$</td>
</tr>
<tr>
<td>$f_A$</td>
<td>$e_{12}$</td>
<td>$&lt;&gt;$</td>
</tr>
<tr>
<td>$f_B$</td>
<td>$e_{13}$</td>
<td>$&lt;&gt;$</td>
</tr>
</tbody>
</table>
## Edge records

<table>
<thead>
<tr>
<th></th>
<th>src</th>
<th>face</th>
<th>next</th>
<th>prev</th>
<th>twin</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_{12}$</td>
<td>$v_1$</td>
<td>$f_A$</td>
<td>$e_{23}$</td>
<td>$e_{31}$</td>
<td>$e_{21}$</td>
</tr>
<tr>
<td>$e_{23}$</td>
<td>$v_2$</td>
<td>$f_A$</td>
<td>$e_{31}$</td>
<td>$e_{12}$</td>
<td>$e_{32}$</td>
</tr>
<tr>
<td>$e_{31}$</td>
<td>$v_3$</td>
<td>$f_A$</td>
<td>$e_{12}$</td>
<td>$e_{23}$</td>
<td>$e_{13}$</td>
</tr>
<tr>
<td>$e_{13}$</td>
<td>$v_1$</td>
<td>$f_A$</td>
<td>??</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$e_{34}$</td>
<td>$v_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$e_{41}$</td>
<td>$v_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Edge records

<table>
<thead>
<tr>
<th></th>
<th>src</th>
<th>face</th>
<th>next</th>
<th>prev</th>
<th>twin</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_{12}$</td>
<td>$v_1$</td>
<td>$f_A$</td>
<td>$e_{23}$</td>
<td>$e_{31}$</td>
<td>$e_{21}$</td>
</tr>
<tr>
<td>$e_{23}$</td>
<td>$v_2$</td>
<td>$f_A$</td>
<td>$e_{31}$</td>
<td>$e_{12}$</td>
<td>$e_{32}$</td>
</tr>
<tr>
<td>$e_{31}$</td>
<td>$v_3$</td>
<td>$f_A$</td>
<td>$e_{12}$</td>
<td>$e_{23}$</td>
<td>$e_{13}$</td>
</tr>
<tr>
<td>$e_{13}$</td>
<td>$v_1$</td>
<td>$f_B$</td>
<td>$e_{34}$</td>
<td>$e_{41}$</td>
<td>$e_{31}$</td>
</tr>
<tr>
<td>$e_{34}$</td>
<td>$v_3$</td>
<td>$f_B$</td>
<td>$e_{41}$</td>
<td>$e_{13}$</td>
<td>$e_{43}$</td>
</tr>
<tr>
<td>$e_{41}$</td>
<td>$v_4$</td>
<td>$f_B$</td>
<td>$e_{13}$</td>
<td>$e_{34}$</td>
<td>$e_{14}$</td>
</tr>
</tbody>
</table>
### Edge records (continued)

<table>
<thead>
<tr>
<th></th>
<th>src</th>
<th>face</th>
<th>next</th>
<th>prev</th>
<th>twin</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_{14}$</td>
<td>$v_1$</td>
<td>$f_U$</td>
<td>$e_{43}$</td>
<td>$e_{21}$</td>
<td>$e_{41}$</td>
</tr>
<tr>
<td>$e_{43}$</td>
<td>$v_4$</td>
<td>$f_U$</td>
<td>$e_{35}$</td>
<td>$e_{14}$</td>
<td>$e_{34}$</td>
</tr>
<tr>
<td>??</td>
<td>??</td>
<td>??</td>
<td>??</td>
<td>??</td>
<td>??</td>
</tr>
<tr>
<td>??</td>
<td>??</td>
<td>??</td>
<td>??</td>
<td>??</td>
<td>??</td>
</tr>
<tr>
<td>$e_{32}$</td>
<td>$v_3$</td>
<td>$f_U$</td>
<td>$e_{21}$</td>
<td>$e_{53}$</td>
<td>$e_{23}$</td>
</tr>
<tr>
<td>$e_{21}$</td>
<td>$v_2$</td>
<td>$f_U$</td>
<td>$e_{14}$</td>
<td>$e_{32}$</td>
<td>$e_{12}$</td>
</tr>
</tbody>
</table>
### Edge records (continued)

<table>
<thead>
<tr>
<th></th>
<th>src</th>
<th>face</th>
<th>next</th>
<th>prev</th>
<th>twin</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_{14}$</td>
<td>$v_1$</td>
<td>$f_U$</td>
<td>$e_{43}$</td>
<td>$e_{21}$</td>
<td>$e_{41}$</td>
</tr>
<tr>
<td>$e_{43}$</td>
<td>$v_4$</td>
<td>$f_U$</td>
<td>$e_{35}$</td>
<td>$e_{14}$</td>
<td>$e_{34}$</td>
</tr>
<tr>
<td>$e_{35}$</td>
<td>$v_3$</td>
<td>$f_U$</td>
<td>$e_{53}$</td>
<td>$e_{43}$</td>
<td>$e_{53}$</td>
</tr>
<tr>
<td>$e_{53}$</td>
<td>$v_5$</td>
<td>$f_U$</td>
<td>$e_{32}$</td>
<td>$e_{35}$</td>
<td>$e_{35}$</td>
</tr>
<tr>
<td>$e_{32}$</td>
<td>$v_3$</td>
<td>$f_U$</td>
<td>$e_{21}$</td>
<td>$e_{53}$</td>
<td>$e_{23}$</td>
</tr>
<tr>
<td>$e_{21}$</td>
<td>$v_2$</td>
<td>$f_U$</td>
<td>$e_{14}$</td>
<td>$e_{32}$</td>
<td>$e_{12}$</td>
</tr>
</tbody>
</table>
Questions...

How to visit all the edges around a vertex, say $v_3$, in counterclockwise order?

<table>
<thead>
<tr>
<th></th>
<th>coord</th>
<th>edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$v_2$</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$v_3$</td>
<td>(0.0, 0.0)</td>
<td>$e_{35}$</td>
</tr>
<tr>
<td>$v_4$</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$v_5$</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Questions... 

How to identify all faces adjacent to a given one, say $f_B$?

<table>
<thead>
<tr>
<th></th>
<th>outer</th>
<th>inner</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_U$</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$f_A$</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$f_B$</td>
<td>$e_{13}$</td>
<td>$&lt;&gt;$</td>
</tr>
</tbody>
</table>

Can the same face be met more than once?
Questions...

How to identify all faces adjacent to a given one, say $f_B$?

<table>
<thead>
<tr>
<th></th>
<th>outer</th>
<th>inner</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_U$</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$f_A$</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$f_B$</td>
<td>$e_{13}$</td>
<td>$&lt;&gt;$</td>
</tr>
</tbody>
</table>

Can the *same* face be met more than once?
Outline

1. DCEL data structure
   - representation
   - example

2. Application of DCELs
   - overlay of two subdivisions
   - plane sweep: edges and vertices
   - plane sweep: faces

3. Further annotations
What does overlay mean?

Overlay of two subdivisions $S_1$ and $S_2$:

A face $f$ belongs to the overlay $Ov(S_1, S_2)$ if and only if $f$ is a maximal connected subset of $f_1 \cap f_2$ for $f_1 \in S_1$ and $f_2 \in S_2$.

Here faces = open sets.
Overlay of two subdivisions $S_1$ and $S_2$:

A face $f$ belongs to the overlay $Ov(S_1, S_2)$ if and only if $f$ is a maximal connected subset of $f_1 \cap f_2$ for $f_1 \in S_1$ and $f_2 \in S_2$.

Here faces = open sets
Overlaying subdivisions: $S_1$
Overlaying subdivisions: $S_2$
Overlaying subdivisions: $S_1 + S_2$
Overlaying subdivisions: $Ov(S_1, S_2)$
Approach

Much information about the edges can be reused

Approach:

- load a copy of the DCELs of $S_1$ and $S_2$ (not a valid DCEL as such)
- process the resulting network of edges and vertices: split edges and link parts together as appropriate
- rebuild face records and assign half-edges’ face fields
Approach

Much information about the edges can be reused

Approach:

- load a copy of the DCELs of $S_1$ and $S_2$
  (not a valid DCEL as such)
- process the resulting network of edges and vertices:
  split edges and link parts together as appropriate
- rebuild face records and assign half-edges’ face fields
Approach

Much information about the edges can be reused

Approach:

- load a copy of the DCELs of $S_1$ and $S_2$ (not a valid DCEL as such)
- process the resulting network of edges and vertices: split edges and link parts together as appropriate
- rebuild face records and assign half-edges’ face fields
Approach

Much information about the edges can be reused

Approach:

- load a copy of the DCELs of $S_1$ and $S_2$
  (not a valid DCEL as such)
- process the resulting network of edges and vertices:
  split edges and link parts together as appropriate
- rebuild face records and assign half-edges’ face fields
Much information about the edges can be reused

Approach:

- load a copy of the DCELs of $S_1$ and $S_2$ (not a valid DCEL as such)
- process the resulting network of edges and vertices: split edges and link parts together as appropriate
- rebuild face records and assign half-edges’ face fields
Plane sweep

Overlay of two subdivisions
plane sweep: edges and vertices
plane sweep: faces
Plane sweep

overlay of two subdivisions
plane sweep: edges and vertices
plane sweep: faces

DCEL data structure
Application of DCELs
Further annotations

C. Mirolo
Plane sweep

- **Invariant:**
  valid DCEL of $\text{Ov}(S_1, S_2)$ on the left of the sweep line
  (except from face information, to be processed later)

- event queue ($EQ$) and sweep line structure ($SL$)
treated as usual...

  - all edges incident at event point $p$ come from
    the same original subdivision: can be used as it is
  - event point $p$ involves items of both subdivisions:
    “tedious but not difficult” local updates
Invariant: 
valid DCEL of $Ov(S_1, S_2)$ on the left of the sweep line  
(except from face information, to be processed later)  

event queue ($EQ$) and sweep line structure ($SL$) 
treated as usual. . .  
- all edges incident at event point $p$ come from 
  the same original subdivision: can be used as it is  
- event point $p$ involves items of both subdivisions: 
  “tedious but not difficult” local updates
Plane sweep

- **Invariant:**
  valid DCEL of $Ov(S_1, S_2)$ on the left of the sweep line (except from face information, to be processed later)

- event queue ($EQ$) and sweep line structure ($SL$) treated as usual...
  - all edges incident at event point $p$ come from the same original subdivision: can be used as it is
  - event point $p$ involves items of both subdivisions: “tedious but not difficult” local updates
Plane sweep

- **Invariant:**
  valid DCEL of $Ov(S_1, S_2)$ on the left of the sweep line (except from face information, to be processed later)

- event queue ($EQ$) and sweep line structure ($SL$) treated as usual...
  - all edges incident at event point $p$ come from the same original subdivision: can be used as it is
  - event point $p$ involves items of both subdivisions: “tedious but not difficult” local updates
Plane sweep

**Invariant:**
valid DCEL of $Ov(S_1, S_2)$ on the left of the sweep line (except from face information, to be processed later)

- event queue ($EQ$) and sweep line structure ($SL$) treated as usual. . .
  - all edges incident at event point $p$ come from the same original subdivision: can be used as it is
  - event point $p$ involves items of both subdivisions: “tedious but not difficult” local updates
Plane sweep

**Invariant:**
valid DCEL of \( Ov(S_1, S_2) \) on the left of the sweep line (except from face information, to be processed later)

- event queue (\( EQ \)) and sweep line structure (\( SL \)) treated as usual...
  - all edges incident at event point \( p \) come from the same original subdivision: can be used as it is
  - event point \( p \) involves items of both subdivisions: “tedious but not difficult” local updates
Plane sweep

- **Invariant:**
  valid DCEL of $Ov(S_1, S_2)$ on the left of the sweep line (except from face information, to be processed later)

- event queue ($EQ$) and sweep line structure ($SL$) treated as usual...
  - all edges incident at event point $p$ come from the same original subdivision: can be used as it is
  - event point $p$ involves items of both subdivisions: “tedious but not difficult” local updates
Which local updates?

DCEL data structure
Application of DCELs
Further annotations

Overlay of two subdivisions
Plane sweep: edges and vertices
Plane sweep: faces

Which local updates?
What must be changed?

<table>
<thead>
<tr>
<th></th>
<th>coord</th>
<th>edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>...</td>
<td>a''</td>
</tr>
<tr>
<td>u'</td>
<td>...</td>
<td>β</td>
</tr>
<tr>
<td>u''</td>
<td>...</td>
<td>e''</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>src . . .</th>
<th>next</th>
<th>prev</th>
<th>twin</th>
</tr>
</thead>
<tbody>
<tr>
<td>e'</td>
<td>u' . . .</td>
<td>ε'</td>
<td>δ'</td>
<td>e''</td>
</tr>
<tr>
<td>e''</td>
<td>u'' . . .</td>
<td>ε''</td>
<td>δ''</td>
<td>e'</td>
</tr>
<tr>
<td>a'</td>
<td>φ . . .</td>
<td>c''</td>
<td>α</td>
<td>a''</td>
</tr>
<tr>
<td>c''</td>
<td>v . . .</td>
<td>γ</td>
<td>a'</td>
<td>c'</td>
</tr>
</tbody>
</table>

overlay of two subdivisions
plane sweep: edges and vertices
plane sweep: faces
And how?

<table>
<thead>
<tr>
<th>coord</th>
<th>edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v$</td>
<td>$a''$</td>
</tr>
<tr>
<td>$u'$</td>
<td>$\beta$</td>
</tr>
<tr>
<td>$u''$</td>
<td>$e''$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>src</th>
<th>...</th>
<th>next</th>
<th>prev</th>
<th>twin</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e'$</td>
<td>$u'$</td>
<td>$\varepsilon'$</td>
<td>$\delta'$</td>
<td>$e''$</td>
</tr>
<tr>
<td>$e''$</td>
<td>$u''$</td>
<td>$\varepsilon''$</td>
<td>$\delta''$</td>
<td>$e'$</td>
</tr>
<tr>
<td>$a'$</td>
<td>$\phi$</td>
<td>$c''$</td>
<td>$\alpha$</td>
<td>$a''$</td>
</tr>
<tr>
<td>$c''$</td>
<td>$v$</td>
<td>$\gamma$</td>
<td>$a'$</td>
<td>$c'$</td>
</tr>
</tbody>
</table>

Overlay of two subdivisions

Plane sweep: edges and vertices

Plane sweep: faces
How to find \textit{next}/\textit{prev}? How much does it cost?

<table>
<thead>
<tr>
<th>coord</th>
<th>edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v )</td>
<td>( a'' )</td>
</tr>
<tr>
<td>( u' )</td>
<td>( \beta )</td>
</tr>
<tr>
<td>( u'' )</td>
<td>( e'' )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>src</th>
<th>next</th>
<th>prev</th>
<th>twin</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e' )</td>
<td>( u' )</td>
<td>( c'' )</td>
<td>( \delta' )</td>
</tr>
<tr>
<td>( e'' )</td>
<td>( u'' )</td>
<td>( b'' )</td>
<td>( \delta'' )</td>
</tr>
<tr>
<td>( a' )</td>
<td>( \phi )</td>
<td>( d' )</td>
<td>( \alpha )</td>
</tr>
<tr>
<td>( c'' )</td>
<td>( v )</td>
<td>( \gamma )</td>
<td>( e' )</td>
</tr>
<tr>
<td>( d' )</td>
<td>( v )</td>
<td>( \varepsilon'' )</td>
<td>( a' )</td>
</tr>
<tr>
<td>( d'' )</td>
<td>( v )</td>
<td>( \varepsilon' )</td>
<td>( c' )</td>
</tr>
</tbody>
</table>
Event types

Events, in summary:

- vertex of $S_1$ only (nothing to do)
- edge of $S_1$ passing through a vertex of $S_2$
- ... and symmetric configurations, of course
- vertex of $S_1$ and $S_2$
- intersection of edges of $S_1$ and $S_2$
Event types

Events, in summary:

- vertex of $S_1$ only (nothing to do)
- edge of $S_1$ passing through a vertex of $S_2$
- ... and symmetric configurations, of course
- vertex of $S_1$ and $S_2$
- intersection of edges of $S_1$ and $S_2$
Event types

Events, in summary:

- vertex of $S_1$ only (nothing to do)
- edge of $S_1$ passing through a vertex of $S_2$
- ... and symmetric configurations, of course
- vertex of $S_1$ and $S_2$
- intersection of edges of $S_1$ and $S_2$
Events, in summary:

- vertex of $S_1$ only (nothing to do)
- edge of $S_1$ passing through a vertex of $S_2$
- ... and symmetric configurations, of course
- vertex of $S_1$ and $S_2$
- intersection of edges of $S_1$ and $S_2$
Event types

Events, in summary:

- vertex of $S_1$ only (nothing to do)
- edge of $S_1$ passing through a vertex of $S_2$
- ... and symmetric configurations, of course
- vertex of $S_1$ and $S_2$
- intersection of edges of $S_1$ and $S_2$
Event types

Events, in summary:

- vertex of $S_1$ only (nothing to do)
- edge of $S_1$ passing through a vertex of $S_2$
- \ldots and symmetric configurations, of course
- vertex of $S_1$ and $S_2$
- intersection of edges of $S_1$ and $S_2$
Vertex $u$ of $Ov(S_1, S_2)$

- local changes: $O(\deg(u))$
- overall: $O( (n + k)\log n)$

$n$ complexity of $S_1 + S_2$; $k$ complexity of $Ov(S_1, S_2)$
Analysis (… so far)

Vertex $u$ of $Ov(S_1, S_2)$ …

- local changes: $O(\text{deg}(u))$

- overall: $O( (n + k)\log n )$

$n$ complexity of $S_1 + S_2$; $k$ complexity of $Ov(S_1, S_2)$
Vertex $u$ of $Ov(S_1, S_2)$ ...

- local changes: $O(\text{deg}(u))$
- overall: $O( (n + k)\log n )$

$n$ complexity of $S_1 + S_2$; $k$ complexity of $Ov(S_1, S_2)$
Analysis (... so far)

Vertex $u$ of $Ov(S_1, S_2)$ ...

- local changes: $O(\deg(u))$
- overall: $O((n + k)\log n)$

$n$ complexity of $S_1 + S_2$; $k$ complexity of $Ov(S_1, S_2)$
New faces: integrations

Faces:
- outer boundary
- list of inner boundaries

Half-edges:
- incident face
- everything else is already computed
New faces: integrations

Faces:
- outer boundary
- list of inner boundaries

Half-edges:
- incident face
- everything else is already computed
New faces: getting information

How many faces?

- as many as the outer boundaries...
- ... + 1 (unbounded face)

How to find the boundary cycles?

- graph: half-edges + next/prev links
- connected components
New faces: getting information

How many faces?

- as many as the outer boundaries…
- … + 1 (unbounded face)

How to find the boundary cycles?

- graph: half-edges + next/prev links
- connected components
How many faces?
- as many as the outer boundaries . . .
- . . . + 1  (unbounded face)

How to find the boundary cycles?
- graph: half-edges + next/prev links
- connected components
How many faces?

- as many as the outer boundaries...
- ... + 1 (unbounded face)

How to find the boundary cycles?

- graph: half-edges + next/prev links
- connected components
New faces: getting information

How to know if a boundary is “outer” or “inner”?  

- follow the direction of half-edges  
- at the (lexicographically) leftmost vertex  
- does next half-edge turn left or right?
New faces: getting information

How to know if a boundary is “outer” or “inner”?

- follow the direction of half-edges
- at the (lexicographically) leftmost vertex
- does next half-edge turn left or right?
New faces: face records

Face records...

- create face records for each outer boundary and for the unbounded face
- set the corresponding field to point to a half-edge of the outer boundary
- set the incident face field accordingly for each half-edge of the outer boundary
New faces: face records

Face records...

- create face records for each outer boundary and for the unbounded face

- set the corresponding field to point to a half-edge of the outer boundary

- set the incident face field accordingly for each half-edge of the outer boundary
New faces: face records

Face records...

- create face records for each outer boundary and for the unbounded face
- set the corresponding field to point to a half-edge of the outer boundary
- set the incident face field accordingly for each half-edge of the outer boundary
New faces: outer vs. inner boundaries

Which disconnected edge cycles bound the same face?

- during plane sweep, register the neighbor edge below the leftmost vertex of each inner boundary cycle
- such items are adjacent along SL!
- The resulting links connect (directly or indirectly) inner and outer boundaries of a same face...
New faces: outer vs. inner boundaries

Which disconnected edge cycles bound the same face?

- during plane sweep, register the neighbor edge below the leftmost vertex of each inner boundary cycle

- such items are adjacent along $SL$!

- The resulting links connect (directly or indirectly) inner and outer boundaries of a same face...
New faces: outer vs. inner boundaries

Which disconnected edge cycles bound the same face?

- during plane sweep, register the neighbor edge below the leftmost vertex of each inner boundary cycle
- such items are adjacent along $SL$!
- The resulting links connect (directly or indirectly) inner and outer boundaries of a same face...
New faces: outer vs. inner boundaries

Which disconnected edge cycles bound the same face?

- during plane sweep, register the neighbor edge below the leftmost vertex of each inner boundary cycle
- such items are adjacent along $SL$!
- The resulting links connect (directly or indirectly) inner and outer boundaries of a same face...
Graph of boundary cycles
Are such links sound and complete?

- The interval on the sweep line between a vertex and the linked edge must belong to just one face, hence both related cycles bound the same face.

- The leftmost vertex of each inner boundary must be linked to some component, which must bound the same face.
Connected components of the graph

Are such links sound and complete?

• The interval on the sweep line between a vertex and the linked edge must belong to just one face, hence both related cycles bound the same face.

• The leftmost vertex of each inner boundary must be linked to some component, which must bound the same face.
Are such links sound and complete?

- The interval on the sweep line between a vertex and the linked edge must belong to just one face, hence both related cycles bound the same face.

- The leftmost vertex of each inner boundary must be linked to some component, which must bound the same face.
Further processing

- Set the incident face field for each half-edge of each inner boundary

- We may want to link each face of the new subdivision with the two overlapping original faces
  (information gathered from plane sweep processing...
Further processing

- Set the incident face field for each half-edge of each inner boundary

- We may want to link each face of the new subdivision with the two overlapping original faces (information gathered from plane sweep processing...)
Further processing

- Set the incident face field for each half-edge of each inner boundary

- We may want to link each face of the new subdivision with the two overlapping original faces (information gathered from plane sweep processing... )
Analysis (overall)

- additional cross pointers to work efficiently

- face information within edge and face records can be set in $O(n + k)$ after the plane sweep

- overall costs sum up to: $O((n + k) \log n)$

where $n = |S_1| + |S_2|$; $k = |Ov(S_1, S_2)|$
Analysis (overall)

- additional cross pointers to work efficiently

- face information within edge and face records can be set in $O(n + k)$ after the plane sweep

Overall costs sum up to: $O((n + k) \log n)$

where $n = |S_1| + |S_2|; \quad k = |Ov(S_1, S_2)|$
Analysis (overall)

- additional cross pointers to work efficiently
- face information within edge and face records can be set in $O(n + k)$ after the plane sweep
- overall costs sum up to: $O((n + k) \log n)$

where $n = |S_1| + |S_2|$; $k = |Ov(S_1, S_2)|$
Outline

1. DCEL data structure
   - representation
   - example

2. Application of DCELs
   - overlay of two subdivisions
   - plane sweep: edges and vertices
   - plane sweep: faces

3. Further annotations
Set operations

As special cases of the above technique:

- union of (simple) polygons
- intersection of polygons
- difference of polygons

all in $O((n + k)\log n)$

results are not always polygons
Set operations

As special cases of the above technique:

- union of (simple) polygons
- intersection of polygons
- difference of polygons

All in \( O( (n + k) \log n ) \)

Results are not always polygons
Set operations

As special cases of the above technique:

- union of (simple) polygons
- intersection of polygons
- difference of polygons

all in \( O((n + k) \log n) \) results are not always polygons
D.E. Muller & F.P. Preparata (1978)
Finding the Intersection of Two Convex Polyhedra
*Theoretical Computer Science,* 7(2)