DCEL in CGAL

Definition

```
typedef  Pm_segment_traits_2<K>         Pm_traits;
typedef  Pm_traits::Point_2             Point;
typedef  Pm_traits::X_monotone_curve_2  Segment;
typedef  Pm_default_dcel<Pm_traits>     DCEL;
typedef  Planar_map_2<DCEL,Pm_traits>   Planar_map;

Planar_map   Ein_geradliniger_planarer_Graph;
```

`Planar_map` – Datentypen

- Iteratoren und Circulatoren

  ```
  Vertex_handle
  Halfedge_handle
  Face_handle
  Vertex_iterator
  Halfedge_iterator
  Face_iterator
  Ccb_halfedge_circulator  (-> Halfedge)
  Halfedge_around_vertex_circulator  (-> Halfedge)
  Holes_iterator  (-> Ccb_halfedge_circulator)
  ```

- `Planar_map::Vertex`

  ```
  Halfedge_around_vertex_circulator  incident_halfedges ()
  bool                               is_incident_edge ( Halfedge_handle e )
  bool                               is_incident_face ( Face_handle f )
  unsigned int                       degree ()
  Halfedge *                         halfedge ()
  Point &                            point ()
  void                               set_point ( const Point & p )
  ```

- `Planar_map::Halfedge`

  ```
  Vertex_handle            source ()
  Vertex_handle            target ()
  Face_handle              face ()
  Halfedge_handle          twin ()
  Halfedge_handle          next_halfedge ()
  Ccb_halfedge_circulator  ccb ()
  Halfedge *               opposite ()
  Halfedge *               next ()
  Vertex *                 vertex ()
  Face *                   face ()
  Segment &                curve ()
  ```

- `Planar_map::Face`

  ```
  bool                     is_unbounded ()
  Holes_iterator           holes_begin ()
  Holes_iterator           holes_end ()
  Halfedge_handle          halfedge_on_outer_ccb ()
  Ccb_halfedge_circulator  outer_ccb ()
  bool                     is_halfedge_on_inner_ccb ( Halfedge_handle e )
  bool                     is_halfedge_on_outer_ccb ( Halfedge_handle e )
  bool                     does_outer_ccb_exist ()
  Halfedge *               halfedge ()
  ```

- Konstanten

```
enum  Locate_type  { VERTEX, EDGE, FACE,
                     UNBOUNDED_VERTEX, UNBOUNDED_EDGE, UNBOUNDED_FACE  }
```

`Planar_map` – Funktionen

- Query

```
Halfedge_handle  locate  ( Point p, Locate_type & lt )
Halfedge_handle  vertical_ray_shoot  ( Point p, Locate_type & lt, bool up_direction )
bool             is_point_in_face  ( Point p, Face_handle f )
```

- Modification

```
Halfedge_handle  insert  ( Segment cv )
Halfedge_handle  split_edge  ( Halfedge_handle e, Segment c1, Segment c2 )
Halfedge_handle  merge_edge  ( Halfedge_handle e1, Halfedge_handle e2, Segment cv )
Face_handle      remove_edge  ( Halfedge_handle e )
```

- Access

```
Vertex_iterator    vertices_begin ()
Vertex_iterator    vertices_end ()
Halfedge_iterator  halfedges_begin ()
Halfedge_iterator  halfedges_end ()
Face_iterator      faces_begin ()
Face_iterator      faces_end ()
Face_handle        unbounded_face ()
```

- Sonstiges

```
bool          is_valid ()
unsigned int  number_of_faces ()
unsigned int  number_of_halfedges ()
unsigned int  number_of_vertices ()
```