Auxiliary Material Overview

Oren Salzman, Tel-Aviv University

TEL AVIV UNIVERSITY

Outline

- (Too short) introduction to CGAL*
- MMS supplied material
- GUI

*slides based on presentation by Erich Berberich in ACG course spring 2009

Disclaimer

- We assume that you are familiar with the notions of
  - C++
  - Inheritance (OOP)
  - C++ Templates (Generics in Java)

Outline

- (Too short) introduction to CGAL*
- MMS supplied material
- GUI

*slides based on presentation by Erich Berberich in ACG course spring 2009

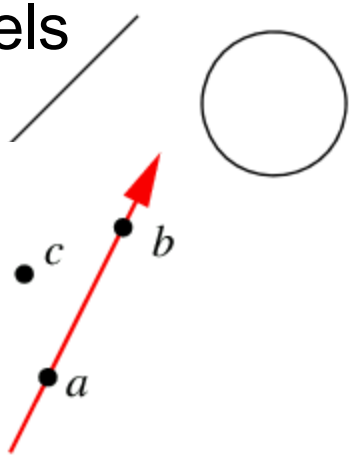
CGAL – Goals

- **Robust geometric computing**
  - Robust (correctness, degeneracies)
  - Efficient (nevertheless: reasonable fast)
  - Ease of use (for users)
  - Homogeneity

- **Geometric computing**
  - Generic programming
  - Exact Geometric Computing Paradigm (by Yap) - all predicates asked by a combinatorial algorithm compute the correct answer

CGAL –Ingredients

- **Implementations of geometric**
  - Objects + Predicates + Constructions , Kernels
  - Algorithms + Data structures
- **Objects :** Points, Lines, Segments, Circles
- **Predicates:** Orientation, Intersections
- **Kernels:** Objects + Predicates + **Number types**
- **Algorithms:** Convex Hull, Triangulations, Minkowski sums



CGAL – Number Types

- Built-in:
  - int, double - fast, inexact
- CGAL
  - Exact: Quotient, MP_Float,
  - Lazy_exact_nt<NT> (first tries an approximation)
  - **Algebraic kernel**
- BOOST:
  - Interval
- GMP
  - Gmpz, Gmpq
- LEDA & CORE
  - Integer, Rational

Generic Programming

- Generic implementations consists of 2 parts:
  - **Instructions** that determine control-flow or updates
  - Set of **requirements** that determine the properties the algorithm's arguments/objects must **satisfy**
    - We call such a set a concept
    - It is abstract, i.e., not working without being instantiated by a model that fulfills the concept

Generic Programming (cont)

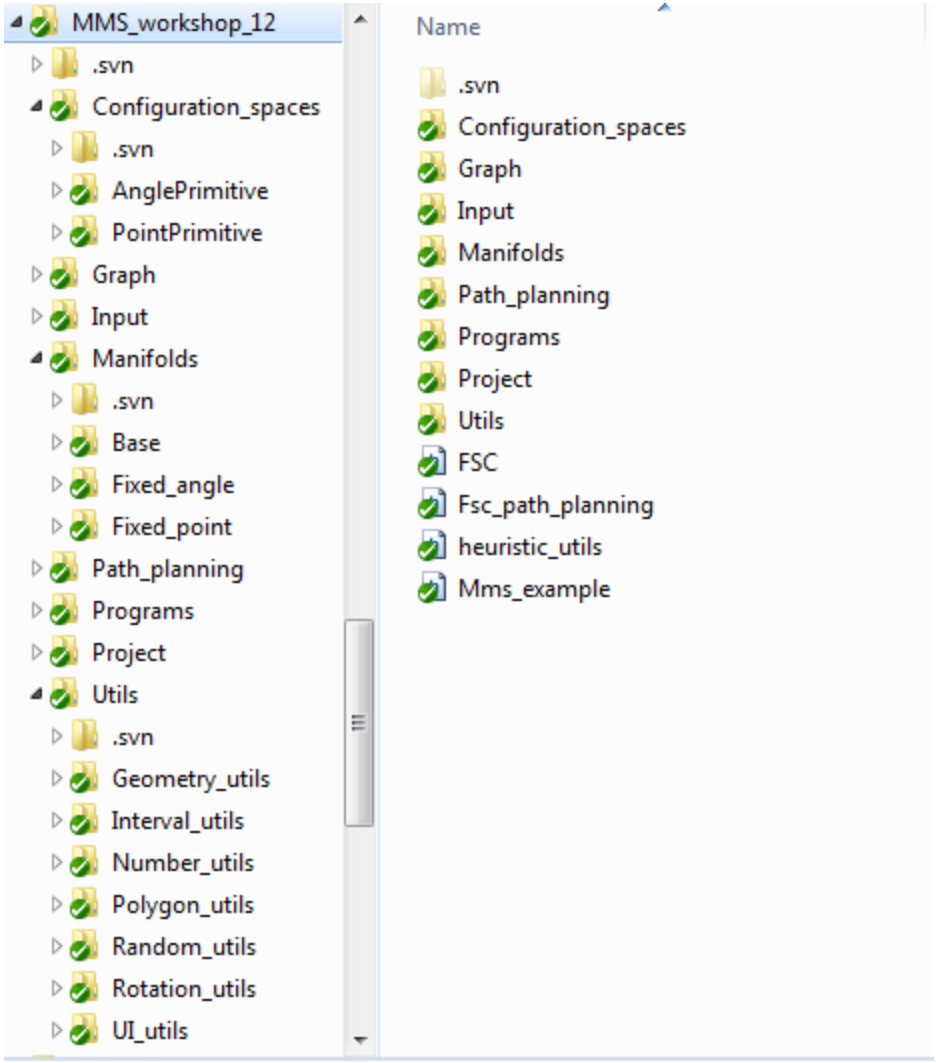
- template <class T>
  void swap(T& a, T& b)
  {
  T tmp = a;
  a = b;
  b = tmp;
  return;
  }
- Argument: type T which must be
  - default constructible
  - Assignable
- Usage:
  int a = 2;
  int b = 4;
  std::swap(a,b)

Outline

- (Too short) introduction to CGAL*
- MMS supplied material
- GUI

*slides based on presentation by Erich Berberich in ACG course spring 2009

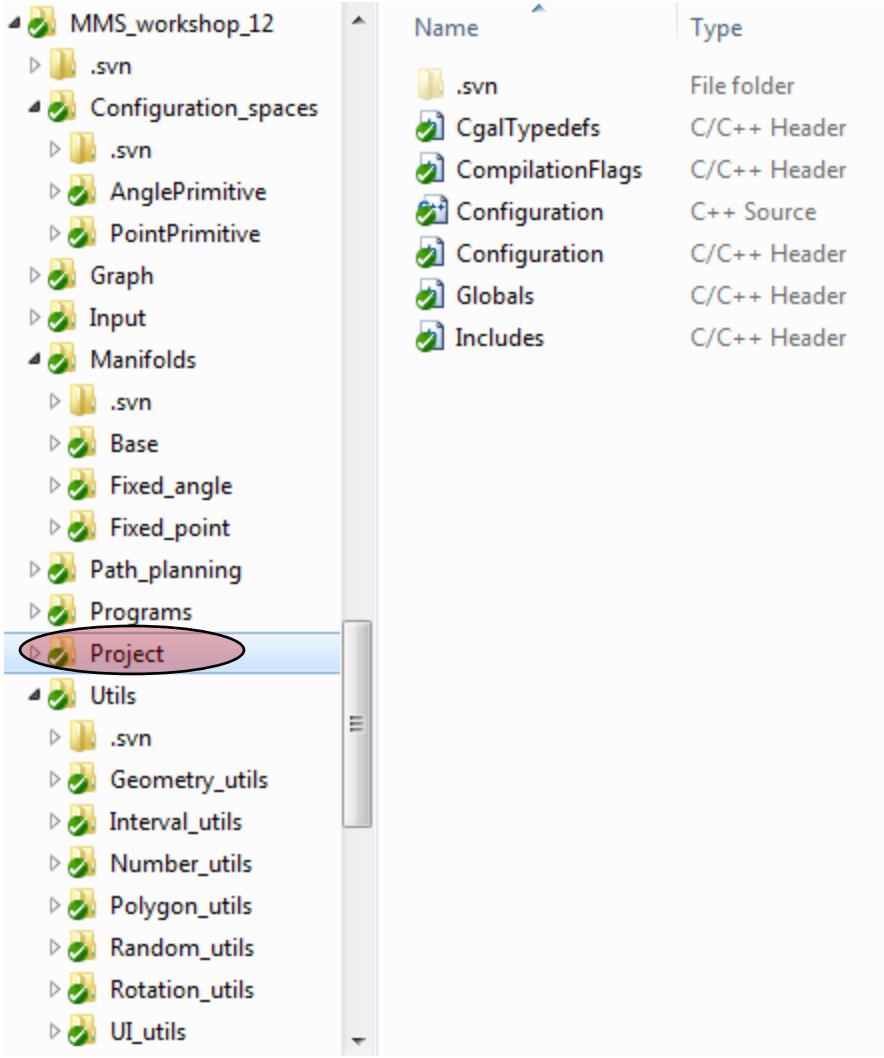
General Contents



General Contents – Project Folder

- General files required for the project.
- There is no need to change these files.

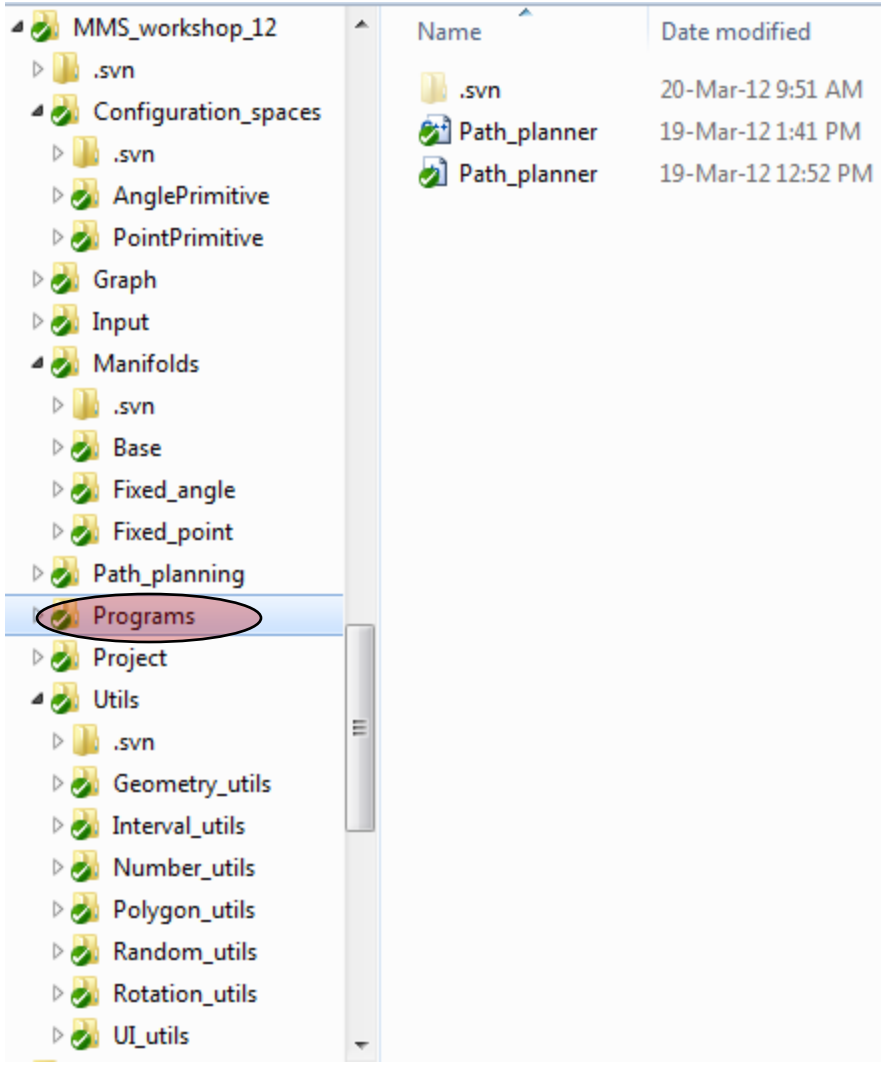
- Includes.h - General include files used by all components
- Globals.h - Global constants
- CompilationFlags.h - Compilation flags
- CgalTypedefs.h - General typedefs
- Configuration.h Configuration.cpp
- Configuration.h, Configuration.cpp - A class that reads and stores a configuration file



General Contents – Programs Folder

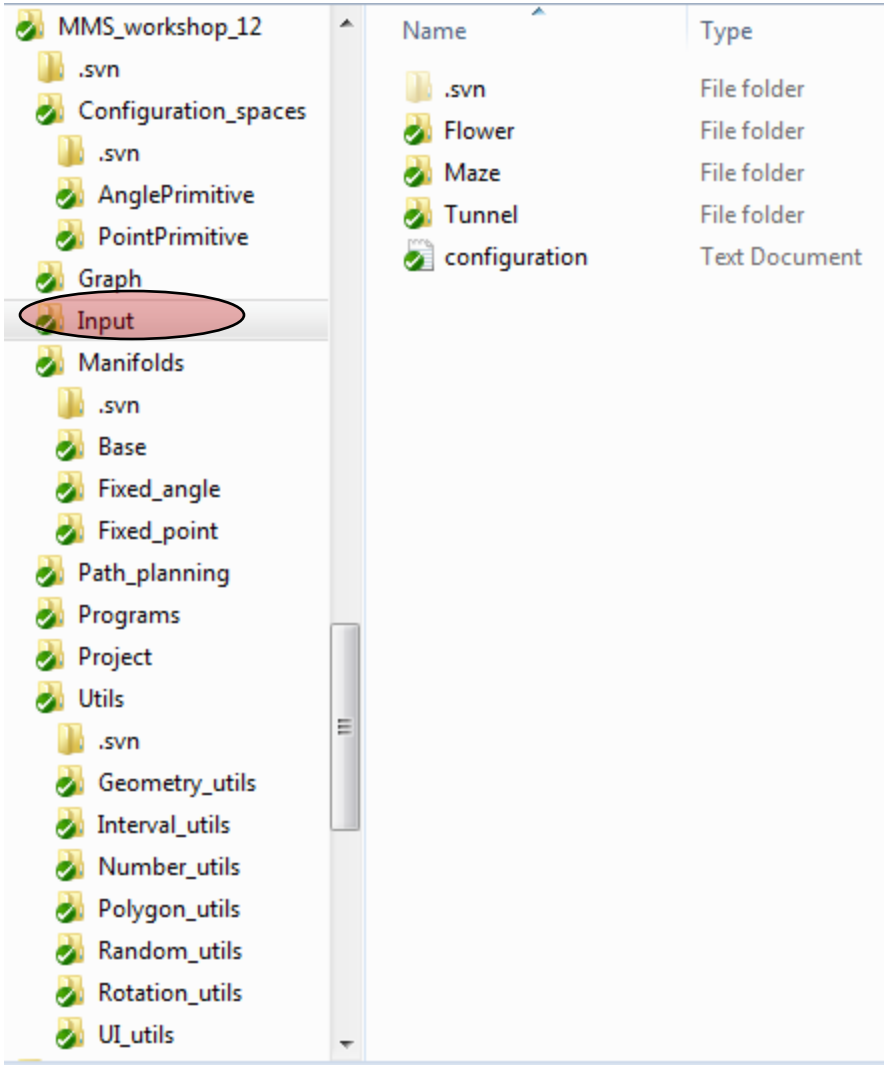
- Contains a simple program that calls an example of the MMS framework
- Declared in Path_planner.h
- Implemented in Path_planner.cpp

- We will cover this example thoroughly



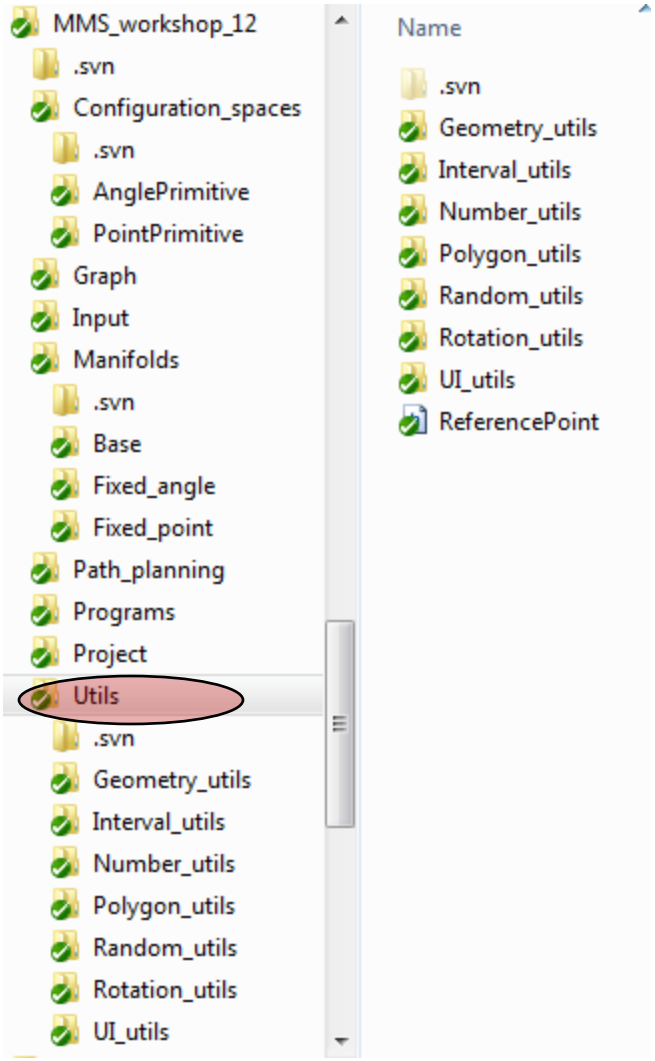
General Contents – Input Folder

- Demo scenarios we will supply
- Configuration file configuration.txt



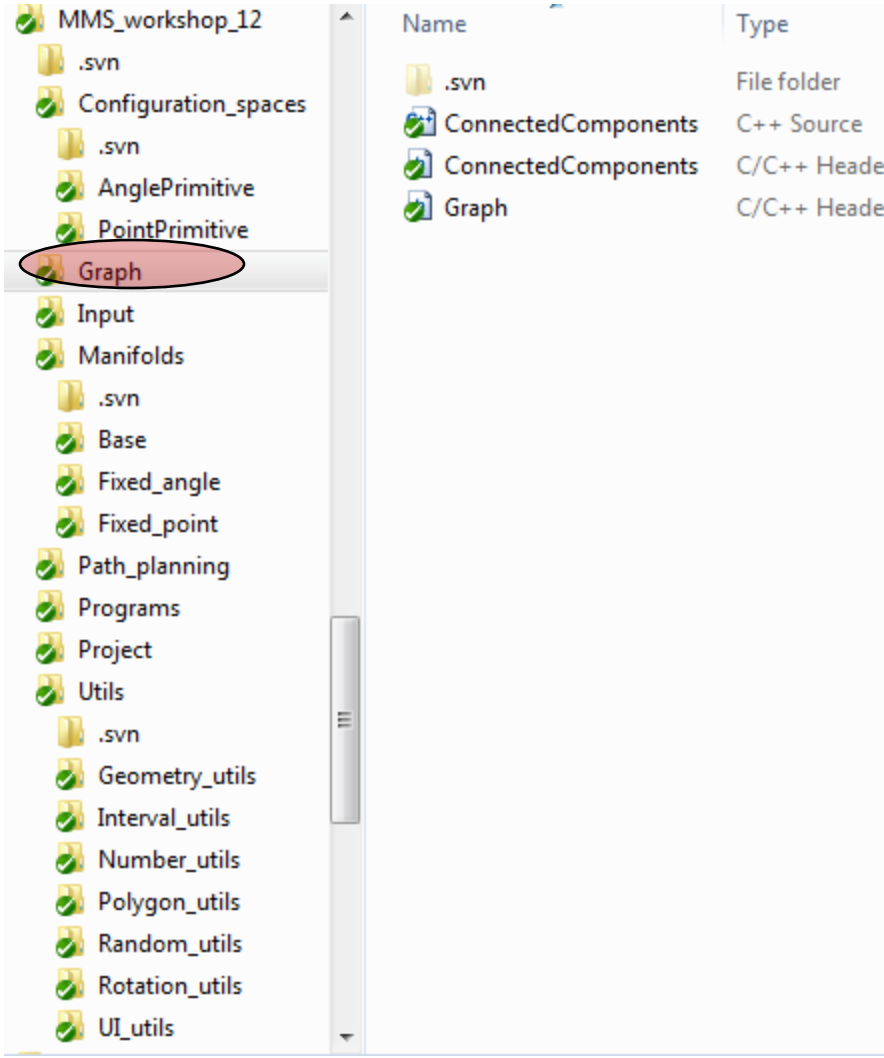
General Contents – Utils Folder

- Includes many utilities used by the MMS framework.
- **Geomtric utilities** (such as bounding volumes and point comparison)
- **Interval utilities** (such as intervals and interval sets)
- **Number type utilities** (such as conversions between algebraic and rational numbers and the approximation of square root numbers),
- **Polygon utilities** (such as intersection predicates, translations and rotations of polygons and more)
- Utilities supporting **random generation** of numbers and geometric objects,
- **Rotation utilities** (such as representing a rational rotation and converting between angles and rotations),



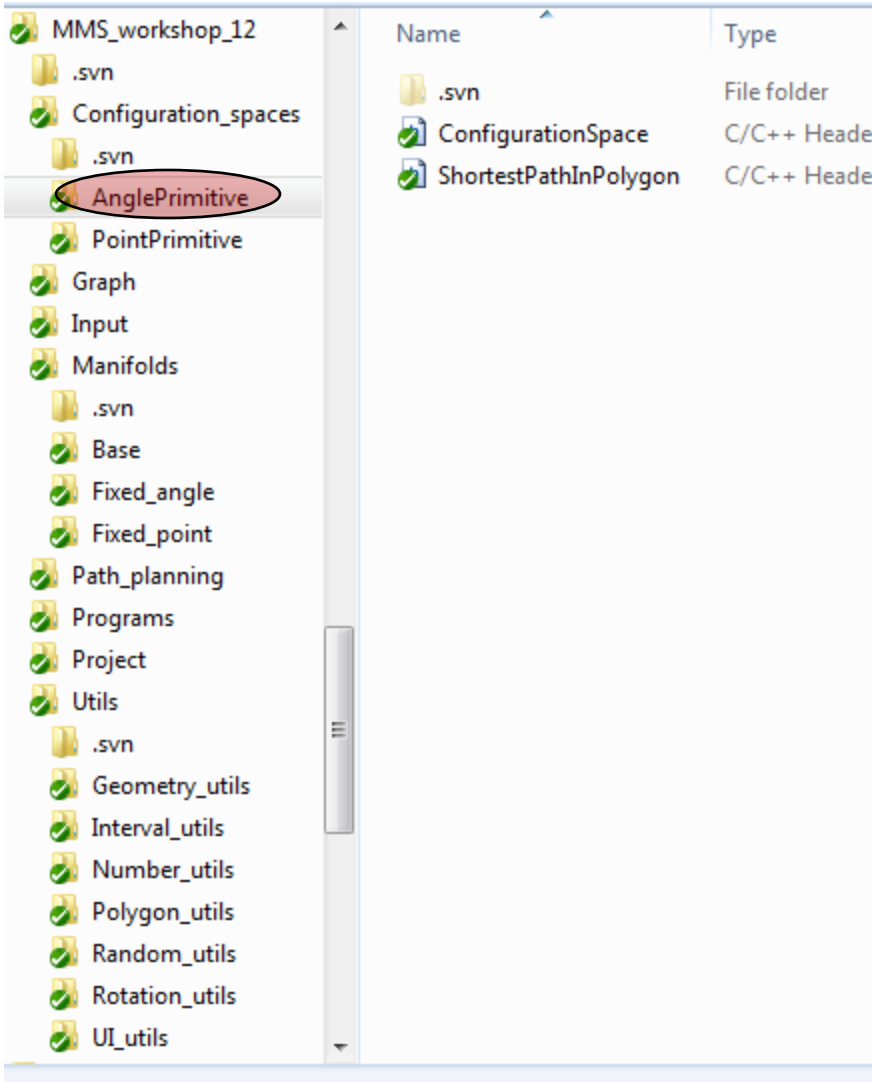
General Contents – Graph Folder

- Graph.h - wrapper class around the boost graph library
- ConnectedComponents.h, ConnectedComponents.cpp - Implementation of queries on the connected components of the graph



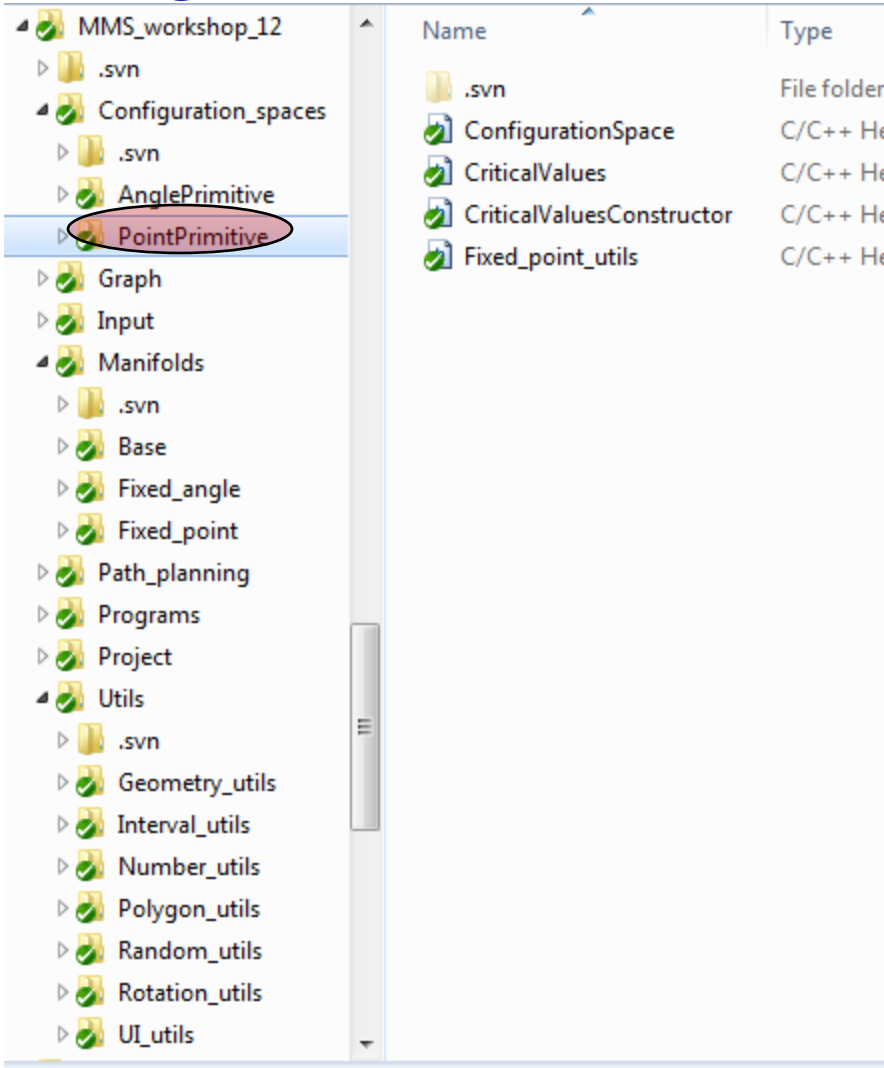
General Contents – Configuration Spaces Folder

- Implementation of configuration spaces for
  - translating robot (AnglePrimitive)
  - rotating robot (PointPrimitive)
- Each folder contains a file named ConfigurationSpace.h with the implementation



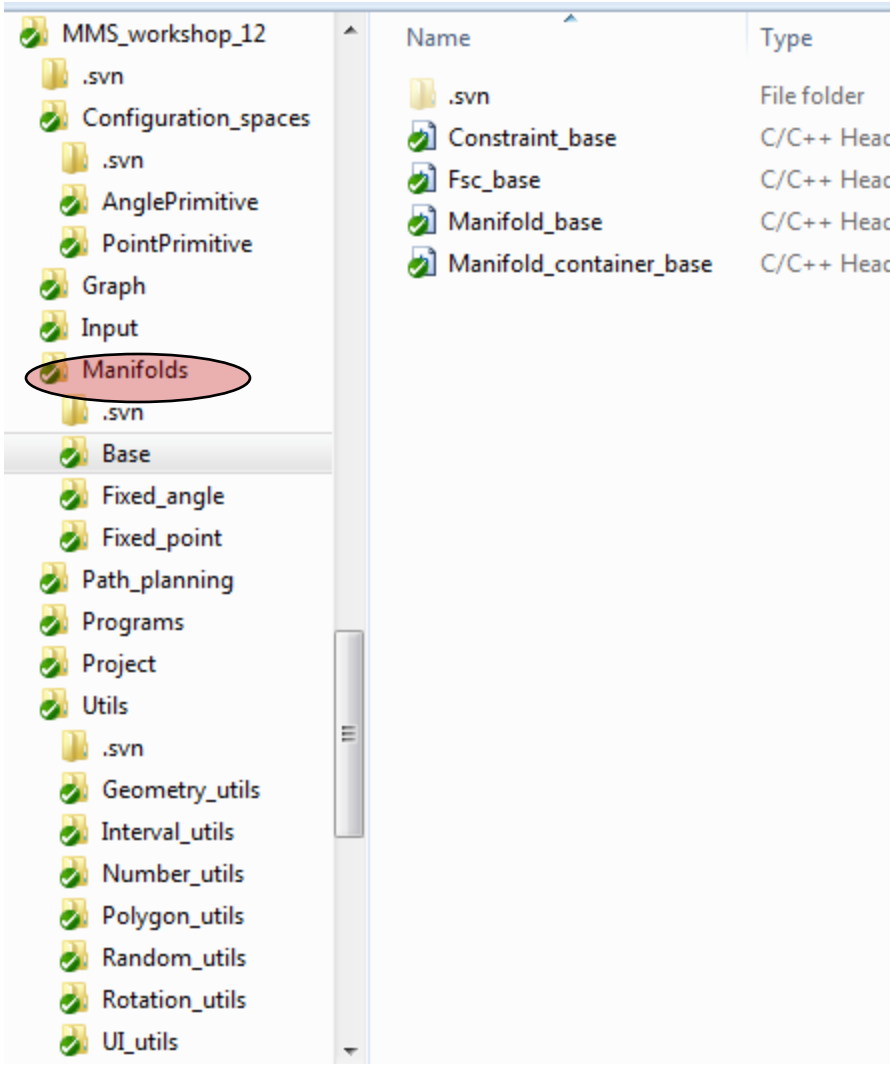
General Contents – Configuration Spaces Folder

- Implementation of configuration spaces for
  - translating robot (AnglePrimitive)
  - rotating robot (PointPrimitive)
- Each folder contains a file named ConfigurationSpace.h with the implementation



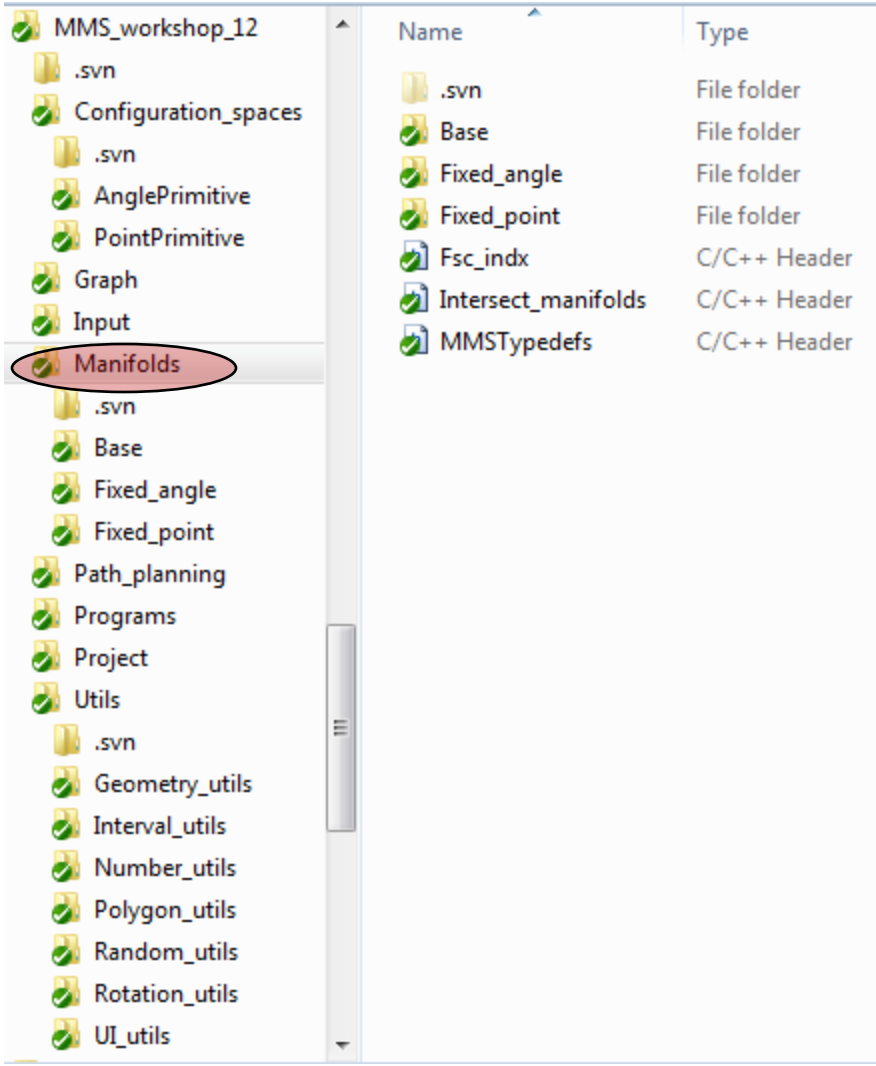
General Contents – Manifolds Folder

- Unified interface for the decomposition of configuration spaces into FSCs.
- Base classes
  - Constraint_base.h – represents a constraint that defines a manifold
  - FSC_base.h – represents a free space cell in the decomposed manifold
  - Manifold_base - represents a container of FSCs
  - Manifold_container_base - represents a container of manifolds



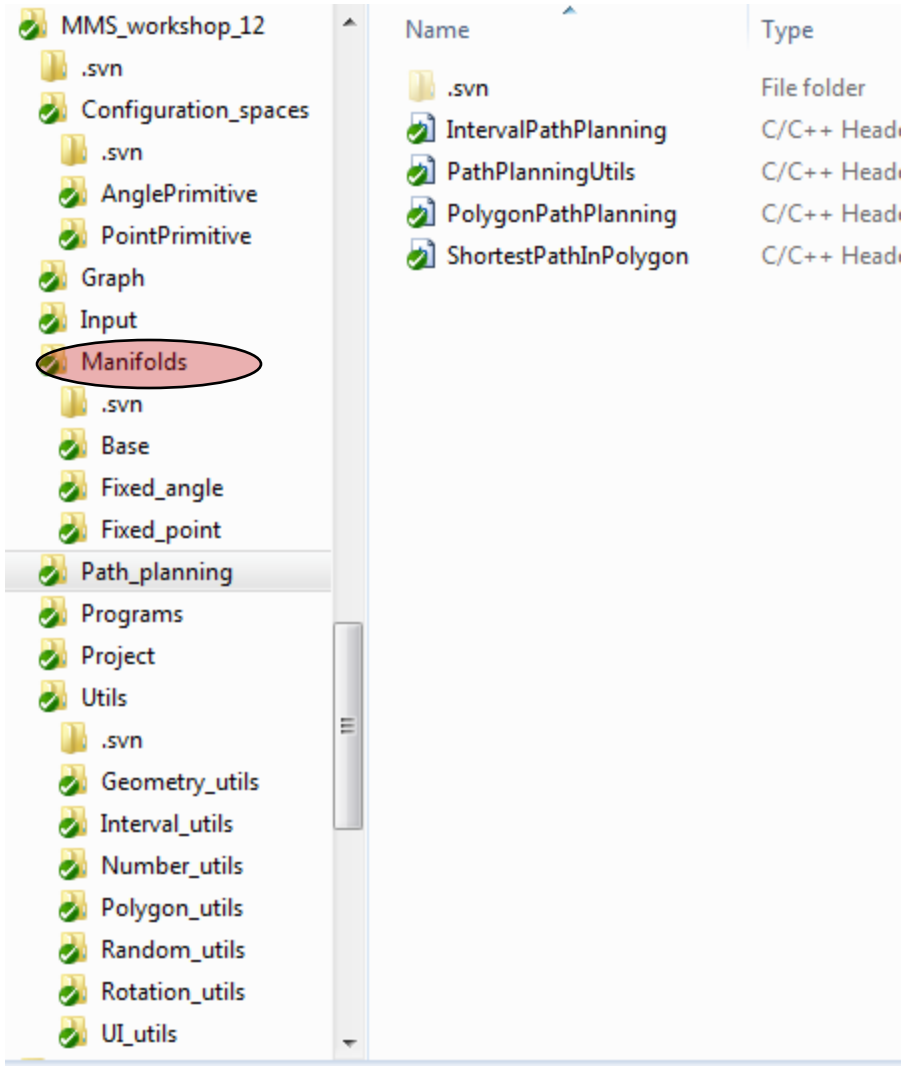
General Contents – Manifolds Folder

- Unified interface for the decomposition of configuration spaces into FSCs.
- Base classes
  - Constraint_base.h – represents a constraint that defines a manifold
  - FSC_base.h – represents a free space cell in the decomposed manifold
  - Manifold_base - represents a container of FSCs
  - Manifold_container_base - represents a container of manifolds
- Inherited classes
- General Files



General Contents – Path Planning Folder

- Local planners in different FSCs.
- Not everything will be supplied but the infrastructure exists



Example - single_robot_planner_example

```
1  Declarations
16 void single_robot_planner_example(int argc, char* argv[])
17 {
18   //typedefs
19   typedef mms::Mms_path_planner_example<>   Planner;
20
21   //loading files from configuration.txt
22   Time_manager tm;
23   tm.write_time_log(std::string("start"));
24
25   Environment<> env(argc,argv);
26   tm.write_time_log(std::string("set environment"));
27
28   //loading scene from environment
29   Planner::Polygon_vec&       workspace(env.get_workspace());
30   Planner::Extended_polygon   my_robot(env.get_robot_a());
31   Planner::Extended_polygon   dynamic_obstacle(env.get_robot_b());
32
33   //in the example we assume that the dynamic obstacle is in the origin
34   dynamic_obstacle.move_origin();
35
36   //add dynamic obstacle as a static obstacle to the workspace and preprocess
37   workspace.push_back(dynamic_obstacle.get_absolute_polygon());
38   Planner planner(workspace, my_robot);
39   planner.preprocess();
40   workspace.pop_back(); //reset workspace
```

Example - single_robot_planner_example

```
42    //construct query
43    Planner::Reference_point q_s, q_t;
44
45    q_s.set_location(Planner::Point(-0.39375,0.39375));
46    q_s.set_rotation(Planner::Rotation(0,1));
47
48    q_t.set_location(Planner::Point(-0.39375,0.21875));
49    q_t.set_rotation(Planner::Rotation(1,0));
50
51    //perform query
52    std::vector<Planner::Reference_point> path;
53    bool found_path = planner.query(q_s, q_t,
54                                    std::back_inserter(path));
55
56    if (!found_path)
57      std::cout<<"no path found :-("<<std::endl;
58    else
59      std::cout<<"path found :-)"<<std::endl;
60
61    return;
62  }
```

Example - class Mms_path_planner_example

```
1   #ifndef MMS_EXAMPLE_H
2   #define MMS_EXAMPLE_H
3
4   #include "Manifolds\MMSTypedefs.h"
5   #include "Manifolds\Fsc_indx.h"
6   #include "Manifolds\Fixed_angle\Fixed_angle_manifold_container.h"
7   #include "Manifolds\Fixed_point\Fixed_point_manifold_container.h"
8   #include "Manifolds\Intersect_manifolds.h"
9   #include "heuristic_utils.h"
10  #include "FSC.h"
11  #include "Graph\Graph.h"
12  #include "Fsc_path_planning.h"
13
14  namespace mms{
15
16  template <typename K = Rational_kernel,
17            typename AK = CGAL::Algebraic_kernel_d_1 <typename CGAL::CORE_arithmetic_kernel::Integer>,
18            typename AK_conversions = Algebraic_kernel_d_1_conversions_rational<typename CGAL::CORE_arithmetic_kernel> >
19  class Mms_path_planner_example { ... };
446 } //mms
447 #endif //MMS_EXAMPLE_H
```

Example - class Mms_path_planner_example

```
19  class Mms_path_planner_example
20  {
21  public:
22    typedef typename K::Point_2                          Point;
23    typedef Rotation<typename K::FT>                     Rotation;
24    typedef typename Reference_point<K>                  Reference_point;
25    typedef Rotation_range_absolute<typename K::FT>      Rotation_range;
26
27    typedef CGAL::Polygon_2 <K>                          Polygon;
28    typedef CGAL::Polygon_with_holes_2<K>                Polygon_with_holes;
29    typedef typename Extended_polygon<K>                 Extended_polygon;
30    typedef typename Smart_polygon_with_holes<K>         Smart_polygon;
31
32    typedef std::vector<typename Polygon>                Polygon_vec;
33    typedef CGAL::Polygon_set_2<K>                       Polygon_set;
34
35  private:
36    typedef Fsc_indx<K>                                  Fsc_indx;
37    typedef FSC<K, AK, AK_conversions>                   Fsc;
38
39    typedef Fixed_angle_manifold_container<K>            Layers;
40    typedef typename Layers::Manifold                    Layer;
41
42    typedef Fixed_point_manifold_container<K, AK, AK_conversions> C_space_lines;
43    typedef typename C_space_lines::Manifold                      C_space_line;
44
45    typedef Graph<Fsc_indx, Less_than_fsc_indx<K> >      Connectiivity_graph;
46    typedef Random_utils<K>                              Random_utils;
```

Example - class Mms_path_planner_example

```
47  private:
48    Polygon_vec&                    _workspace;
49    Polygon_vec                     _decomposed_workspace;
50    Extended_polygon&               _robot;
51
52    Connectiivity_graph             _graph;
53
54    std::vector<Rotation>           _rotations;
55    Layers                          _layers;
56    C_space_lines                   _lines;
57
58    Random_utils                    _rand;
59    AK                              _ak;
```

Example - class Mms_path_planner_example

```
60  public:
61    //constructor
62    Mms_path_planner_example  (Polygon_vec &workspace, Extended_polygon& robot) { ... }
66    //preprocess
67    void preprocess (const unsigned int num_of_angles = configuration.get_slices_granularity()) { ... }
82    //query
83    template <typename OutputIterator>
84    bool query( const Reference_point& source, const Reference_point& target,
85                OutputIterator& oi) { ... }
154 private: //layer methods
155   void generate_rotations(const unsigned int num_of_angles) { ... }
180   void add_layer(const Rotation& rotation) { ... }
190 private:
191   void generate_connectors() { ... }
195   void generate_connectors_random() { ... }
200   void generate_connector_random() { ... }
258 private: //filtering methods
259   bool filter_out(typename C_space_line::Constraint& constraint) { ... }
290 private: //Connectiivity_graph methods
291   void update_connectivity_graph_vertices(Layer& layer, int layer_id) { ... }
300   void update_connectivity_graph(int c_space_line_id) { ... }
329 private: //query related methods
330   template <typename OutputIterator>
331   Reference_point connect_to_graph( const Reference_point& ref_p,
332                                     OutputIterator& oi) { ... }
375 private: //Fsc_indx related methods
376   Fsc_indx get_containig_fsc(const Reference_point& ref_p) { ... }
392   Fsc* get_fsc(const Fsc_indx& fsc_indx) { ... }
410   Reference_point  get_intersection(const Fsc_indx& fsc_indx_1, const Fsc_indx& fsc_indx_2) { ... }
430 private:  //caching related methods
431   void decompose_workspace_into_convex_polygons() { ... }
436   template <typename OutputIterator>
437   void decompose_into_convex_polygons(const Polygon& polygon, OutputIterator& oi) { ... }
444 };  //Mms_path_planner_example
445
446 } //mms
```

Example - class Mms_path_planner_example

```
67  void preprocess  (const unsigned int num_of_angles = configuration.get_slices_granularity())
68  {
69    generate_rotations(num_of_angles);
70    decompose_workspace_into_convex_polygons();
71
72    BOOST_FOREACH (Rotation rotation, _rotations)
73     add_layer(rotation);
74    global_tm.write_time_log(std::string("finished layers"));
75
76    generate_connectors();
77    global_tm.write_time_log(std::string("finished connectors"));
78
79    global_tm.write_time_log(std::string("finished preproccesing"));
80    return;
81  }
```

Example - class Mms_path_planner_example

```
67  void preprocess  (const unsigned int num_of_angles = configuration.get_slices_granularity())
68  {
69    generate_rotations(num_of_angles);
70    decompose_workspace_into_convex_polygons();
71
72    BOOST_FOREACH (Rotation rotation, _rotations)
73     add_layer(rotation);
74    global_tm.write_time_log(std::string("finished layers"));
75
76    generate_connectors();
77    global_tm.write_time_log(std::string("finished connectors"));
78
79    global_tm.write_time_log(std::string("finished preproccesing"));
80    return;
81  }
```

```
180 void add_layer(const Rotation& rotation)
181 {
182   //create layer
183   Layer* layer_ptr = new Layer (Layer::Constraint(rotation));
184   layer_ptr->decompose(_robot, _decomposed_workspace);
185   int layer_id = _layers.add_manifold(layer_ptr);
186
187   update_connectivity_graph_vertices(*layer_ptr, layer_id);
188   return;
189 }
```

Example - class Mms_path_planner_example

```
200  void generate_connector_random()
201  {
202    ////////////////////////////////////////////////////////////////
203    //get free point in the configuration space on one of the layers
204    ////////////////////////////////////////////////////////////////

213    C_space_line* c_space_line_ptr;
214    C_space_line::Constraint constraint;
215    ////////////////////////////////////////////////////////////////
216    //choose roi
217    ////////////////////////////////////////////////////////////////


238    ////////////////////////////////////////////////////////////////
239    //attempt to filter
240    ////////////////////////////////////////////////////////////////
241    if (filter_out(constraint))
242      return;
243
244    ////////////////////////////////////////////////////////////////
245    //create connector
246    ////////////////////////////////////////////////////////////////
247    c_space_line_ptr = new C_space_line (constraint, _ak);
248    c_space_line_ptr->decompose(_robot, _decomposed_workspace);
249    int c_space_line_id = _lines.add_manifold(c_space_line_ptr);
250
251    ////////////////////////////////////////////////////////////////
252    //update connectivity graph
253    ////////////////////////////////////////////////////////////////
254    update_connectivity_graph(c_space_line_id);
255    return;
256
257  }
```

Example - class Mms_path_planner_example

```
83    template <typename OutputIterator>
84    bool query( const Reference_point& source, const Reference_point& target,
85                OutputIterator& oi)
86    {
87      ////////////////////////////////////
88      //connect source and target to graph
89      ////////////////////////////////////
90      std::vector<Reference_point> source_path, target_path;
91      Reference_point perturbed_source = connect_to_graph(source, std::back_inserter(source_path));
92      Reference_point perturbed_target = connect_to_graph(target, std::back_inserter(target_path));
93
94      if (source_path.empty() || target_path.empty() )
95        return false;
96
97      ////////////////////////////////////
98      //find path of fscs(if exists)
99      ////////////////////////////////////
100     Fsc_indx source_fsc_indx (get_containig_fsc(perturbed_source));
101     CGAL_postcondition (source_fsc_indx != Fsc_indx());
102     Fsc_indx target_fsc_indx (get_containig_fsc(perturbed_target));
103     CGAL_postcondition (target_fsc_indx != Fsc_indx());
104
105     std::list<Fsc_indx> fsc_indx_path;
106     if (source_fsc_indx == target_fsc_indx)
107       fsc_indx_path.push_back(source_fsc_indx);
108     else
109       _graph.find_path(source_fsc_indx, target_fsc_indx, fsc_indx_path);
110
111     if (fsc_indx_path.empty())
112       return false;
```

Example - class Mms_path_planner_example

```
114     ////////////////////////////////////
115     //find path of configurations
116     ////////////////////////////////////
117     BOOST_FOREACH(Reference_point ref_p, source_path)
118       *oi++ = ref_p;
119
120     int             curr_fsc_indx = 0;
121     Reference_point curr_ref_p    = perturbed_source;
122     std::list<Fsc_indx>::iterator curr, next;
123     next = curr = fsc_indx_path.begin();
124     ++next;
125     while (next != fsc_indx_path.end())
126     {
127       Reference_point  next_ref_p = get_intersection(*curr, *next);
128
129       Fsc* fsc_ptr = get_fsc(*curr);
130       CGAL_postcondition(fsc_ptr->contains(curr_ref_p) && fsc_ptr->contains(next_ref_p));
131
132       plan_path(fsc_ptr, curr_ref_p, next_ref_p, oi);
133       curr++;      next++;
134       curr_ref_p = next_ref_p;
135       delete fsc_ptr;
136     }
137
138     Fsc* fsc_ptr = get_fsc(*curr);
139     plan_path(fsc_ptr, curr_ref_p, perturbed_target, oi);
140     delete fsc_ptr;
141
142     BOOST_FOREACH(Reference_point ref_p, target_path)
143       *oi++ = ref_p;
144     return true;
```

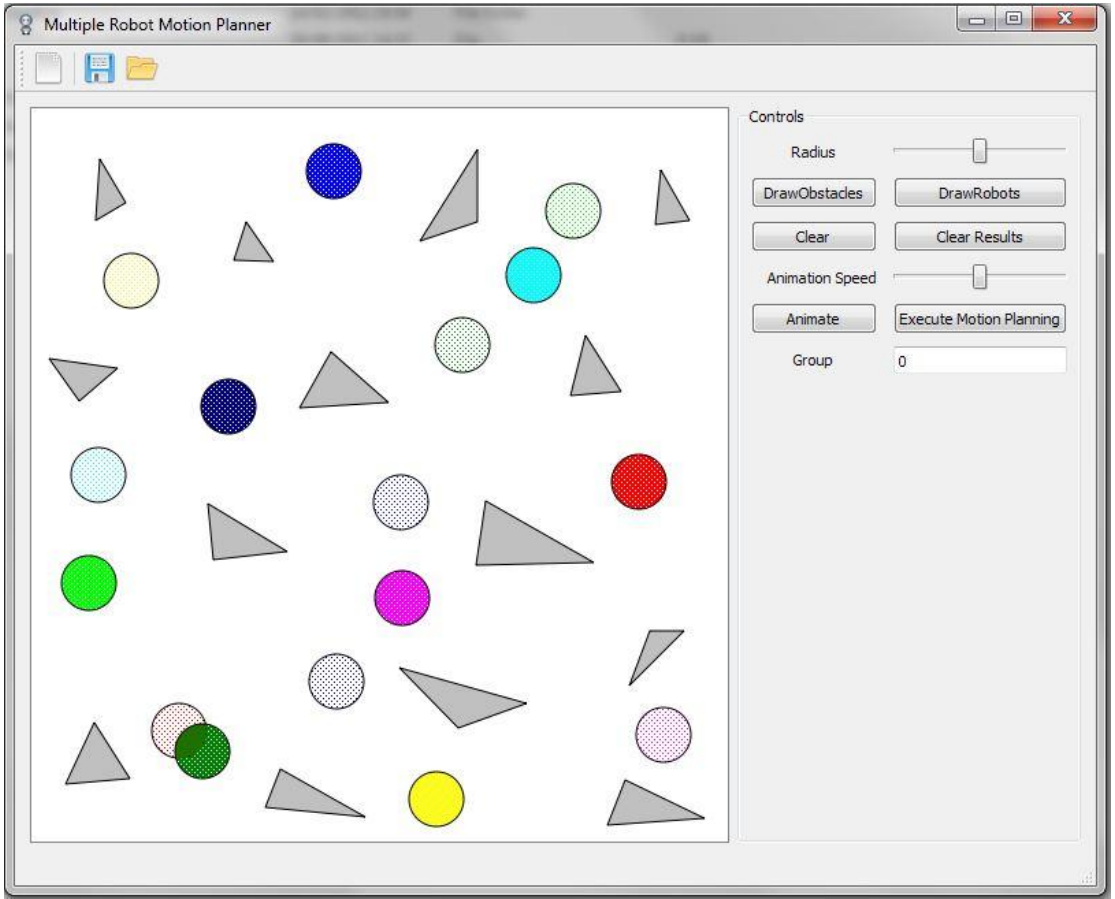
Outline

- (Too short) introduction to CGAL*
- MMS supplied material
- GUI

*slides based on presentation by Erich Berberich in ACG course spring 2009

GUI

- Scene generation
- Path visualization



Tips

- Use precompiled headers
- CGAL Manual