Auxiliary Material Overview

Oren Salzman, 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 consist 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

![Image showing a directory structure]

- MMS_workshop_12
  - .svn
  - Configuration_spaces
    - .svn
    - AnglePrimitive
    - PointPrimitive
  - Graph
  - Input
  - Manifolds
    - .svn
    - Base
    - Fixed_angle
    - Fixed_point
  - Path_planning
  - Programs
  - Project
  - Utils
    - .svn
    - Geometry_utils
    - Interval_utils
    - Number_utils
    - Polygon_utils
    - Random_utils
    - Rotation_utils
    - UI_utils
  - FSC
  - Fsc_path_planning
  - heuristic_utils
  - Mms_example
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
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

```c
[Declarations]

void single_robot_planner_example(int argc, char* argv[])
{
  //typedef
  typedef mms::Mms_path_planner_example<> Planner;
  //loading files from configuration.txt
  Time_manager tm;
  tm.write_time_log(std::string("start"));

  Environment<> env(argc, argv);
  tm.write_time_log(std::string("set environment"));

  //loading scene from environment
  Planner::Polygon_vec& workspace(env.get_workspace());
  Planner::Extended_polygon my_robot(env.get_robot_a());
  Planner::Extended_polygon dynamic_obstacle(env.get_robot_b());

  //in the example we assume that the dynamic obstacle is in the origin
  dynamic_obstacle.move_origin();

  //add dynamic obstacle as a static obstacle to the workspace and preprocess
  workspace.push_back(dynamic_obstacle.get_absolute_polygon());
  Planner planner(workspace, my_robot);
  planner.preprocess();
  workspace.pop_back(); //reset workspace
```
Example - single_robot_planner_example

```
//construct query
Planner::Reference_point q_s, q_t;
q_s.set_location(Planner::Point(-0.39375, 0.39375));
q_s.set_rotation(Planner::Rotation(0, 1));
q_t.set_location(Planner::Point(-0.39375, 0.21875));
q_t.set_rotation(Planner::Rotation(1, 0));

//perform query
std::vector<Planner::Reference_point> path;
bool found_path = planner.query(q_s, q_t,
    std::back_inserter(path));

if (!found_path)
    std::cout << "no path found :-(" << std::endl;
else
    std::cout << "path found :-)" << std::endl;
return;
```
Example - class Mms_path_planner_example
Example - class Mms_path_planner_example

class Mms_path_planner_example
{
  public:
    typedef typename K::Point_2 Point;
    typedef Rotation<typename K::FT> Rotation;
    typedef typename Reference_point<K> Reference_point;
    typedef Rotation_range_absolute<typename K::FT> Rotation_range;
    typedef CGAL::Polygon_2 <K> Polygon;
    typedef CGAL::Polygon_with_holes_2<K> Polygon_with_holes;
    typedef typename Extended_polygon<K> Extended_polygon;
    typedef typename Smart_polygon_with_holes<K> Smart_polygon;
    typedef std::vector<typename Polygon> Polygon_vec;
    typedef CGAL::Polygon_set_2<K> Polygon_set;

  private:
    typedef Fsc_idx<K> Fsc_idx;
    typedef FSC<K, AK, AK_conversions> Fsc;
    typedef Fixed_angle_manifold_container<K> Layers;
    typedef typename Layers::Manifold Layer;
    typedef Fixed_point_manifold_container<K, AK, AK_conversions> C_space_lines;
    typedef typename C_space_lines::Manifold C_space_line;
    typedef Graph<Fsc_idx, Less_than_fsc_idx<K>> Connectivity_graph;
    typedef Random_utils<K> Random_utils;
}
Example - class Mms_path_planner_example

```cpp
private:
    Polygon_vec& _workspace;
    Polygon_vec _decomposed_workspace;
    Extended_polygon& _robot;
    Connectivity_graph _graph;
    std::vector<Rotation> rotations;
    Layers _layers;
    C_space_lines _lines;
    Random_utils _rand;
    AK _ak;
```
Example - class Mms_path_planner_example

```cpp
public:

    //constructor
    Mms_path_planner_example (Polygon_vec &workspace, Extended_polygon& robot) { ... }

    //preprocess
    void preprocess (const unsigned int num_of_angles = configuration.get_slices_granularity()) { ... }

    //query
    template <typename OutputIterator>
    bool query( const Reference_point& source, const Reference_point& target,
                OutputIterator& oi) { ... }

private:  //layer methods

    void generate_rotations(const unsigned int num_of_angles) { ... }
    void add_layer(const Rotation& rotation) { ... }
    void generate_connectors() { ... }
    void generate_connectors_random() { ... }
    void generate_connector_random() { ... }

private:  //filtering methods

    bool filter_out(typename C_space_line::Constraint& constraint) { ... }

private:  //Connectivity_graph methods

    void update_connectivity_graph_vertices(Layer& layer, int layer_id) { ... }
    void update_connectivity_graph(int c_space_line_id) { ... }

private:  //query related methods

    template <typename OutputIterator>
    Reference_point connect_to_graph( const Reference_point& ref_p,
                                       OutputIterator& oi) { ... }

private:  //Fsc_index related methods

    Fsc_index get_containing_fsc(const Reference_point& ref_p) { ... }
    Fsc* get_fsc(const Fsc_index& fsc_index) { ... }
    Reference_point get_intersection(const Fsc_index& fsc_index_1, const Fsc_index& fsc_index_2) { ... }

private:  //caching related methods

    void decompose_workspace_into_convex_polygons() { ... }
    template <typename OutputIterator>
    void decompose_into_convex_polygons(const Polygon& polygon, OutputIterator& oi) { ... }

};  //Mms_path_planner_example
```
Example - class Mms_path_planner_example

```cpp
void preprocess (const unsigned int num_of_angles = configuration.get_slicesGranularity())
{
  generate_rotations(num_of_angles);
  decompose_workspace_into_convex_polygons();

  BOOST_FOREACH (Rotation rotation, _rotations)
    add_layer(rotation);
  global_tm.write_time_log(std::string("finished layers"));

  generate_connectors();
  global_tm.write_time_log(std::string("finished connectors"));

  global_tm.write_time_log(std::string("finished preprocesing"));
  return;
}
```
Example - class Mms_path_planner_example

```cpp
void preprocess (const unsigned int num_of_angles = configuration.get_slices_granularity())
{
    generate_rotations(num_of_angles);
    decompose_workspace_into_convex_polygons();

    BOOST_FOREACH (Rotation rotation, _rotations)
        add_layer(rotation);
    global_tm.write_time_log(std::string("finished layers"));

    generate_connectors();
    global_tm.write_time_log(std::string("finished connectors"));
    global_tm.write_time_log(std::string("finished preprocessing"));
    return;
}

void add_layer (const Rotation& rotation)
{
    //create layer
    Layer* layer_ptr = new Layer (Layer::Constraint(rotation));
    layer_ptr->decompose(_robot, _decomposed_workspace);
    int layer_id = _layers.add_manifold(layer_ptr);
    update_connectivity_graph_vertices(*layer_ptr, layer_id);
    return;
}
```
Example - class Mms_path_planner_example

```cpp
void generate_connector_random()
{
    //get free point in the configuration space on one of the layers

    C_space_line* c_space_line_ptr;
    C_space_line::Constraint constraint;
    //choose roi

    //attempt to filter
    if (filter_out(constraint))
        return;

    //create connector
    c_space_line_ptr = new C_space_line (constraint, _ak);
    c_space_line_ptr->decompose(_robot, _decomposed_workspace);
    int c_space_line_id = _lines.add_manifold(c_space_line_ptr);

    //update connectivity graph
    update_connectivity_graph(c_space_line_id);
    return;
}
```
```cpp
template<typename OutputIterator>
bool query(const Reference_point& source, const Reference_point& target, OutputIterator& oi)
{
    /////////////////////////////////////////////////////////////////////////////////
    //connect source and target to graph
    /////////////////////////////////////////////////////////////////////////////////
    std::vector<Reference_point> source_path, target_path;
    Reference_point perturbed_source = connect_to_graph(source, std::back_inserter(source_path));
    Reference_point perturbed_target = connect_to_graph(target, std::back_inserter(target_path));

    if (source_path.empty() || target_path.empty())
        return false;

    /////////////////////////////////////////////////////////////////////////////////
    //find path of fscs(if exists)
    /////////////////////////////////////////////////////////////////////////////////
    Fsc_index source_fsc_index = get_containing_fsc(perturbed_source);
    CGAL_postcondition(source_fsc_index != Fsc_index());
    Fsc_index target_fsc_index = get_containing_fsc(perturbed_target);
    CGAL_postcondition(target_fsc_index != Fsc_index());

    std::list<Fsc_index> fsc_index_path;
    if (source_fsc_index == target_fsc_index)
        fsc_index_path.push_back(source_fsc_index);
    else
        _graph.find_path(source_fsc_index, target_fsc_index, fsc_index_path);

    if (fsc_index_path.empty())
        return false;
```
Example - class Mms_path_planner_example

```cpp
#include <algorithm>
#include <boost/algorithm/four_of_each.hpp>
#include <boost/foreach.hpp>

// find path of configurations
BOOST_FOREACH(Reference_point ref_p, source_path)
  *oi++ = ref_p;

int curr_fsc_idx = 0;
Reference_point curr_ref_p = perturbed_source;
std::list<Fsc_idx>::iterator curr, next;
next = curr = fsc_idx_path.begin();
++next;
while (next != fsc_idx_path.end())
{
  Reference_point next_ref_p = get_intersection(*curr, *next);
  Fsc* fsc_ptr = get_fsc(*curr);
  CGAL_postcondition(fsc_ptr->contains(curr_ref_p) && fsc_ptr->contains(next_ref_p));
  plan_path(fsc_ptr, curr_ref_p, next_ref_p, oi);
  curr++; next++;
  curr_ref_p = next_ref_p;
  delete fsc_ptr;
}

Fsc* fsc_ptr = get_fsc(*curr);
plan_path(fsc_ptr, curr_ref_p, perturbed_target, oi);
delete fsc_ptr;
BOOST_FOREACH(Reference_point ref_p, target_path)
  *oi++ = ref_p;
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