Getting Started with OOPSMP

Mark Moll
Physical & Biological Computing Group
Rice University
Houston, TX
USA

Outline

- Motivation
- Main features of OOPSMP
- Visual walk-through of using OOPSMP
- Detailed walk-through
- Concluding remarks

Motivation

The need for a motion planning program

- Teaching tool:
  - Play with different algorithms
  - Quickly create interesting motion planning problems
  - Minimal programming overhead

The need for a motion planning program

- Research tool:
  - Easily compare new algorithms with “standard” algorithms
  - Run algorithms on benchmark problems
  - Reuse common data structures

Other motion planning software

- **MPK**, Schwarzer, Saha, Latombe
- **MSL**, LaValle et al.
- **OpenRAVE**, Diankov & Kuffner
- **KineoWorks**, Laumond et al.

Other robotics software

- Player/Stage/Gazebo
- Microsoft Robotics Studio
- Willow Garage ROS
- Carmen

OOPSMP:

Object-Oriented Programming System for Motion Planning

OOPSMP:

Object-Oriented Programming System for Motion Planning

There is no other program that:

- contains several motion planning algorithms
- is easily extensible with new algorithms
- has an easy to use graphical front-end
- runs on a variety of platforms

OOPSMP: main features

- Efficient and robust implementations of
  - motion planning methods and components
  - general data structures and utilities
- Plug-and-play functionality
  - easy to extend
- Can be used as a library within, say, a game engine or embedded robotics system

OOPSMP: main features (contd.)

- Stand-alone version
  - command line version
  - graphical version
- Google SketchUp plugin
  - easy to create motion planning problems
- Source code available for download
- Runs on Windows, Linux, and Apple OS X

OOPSMP workflow

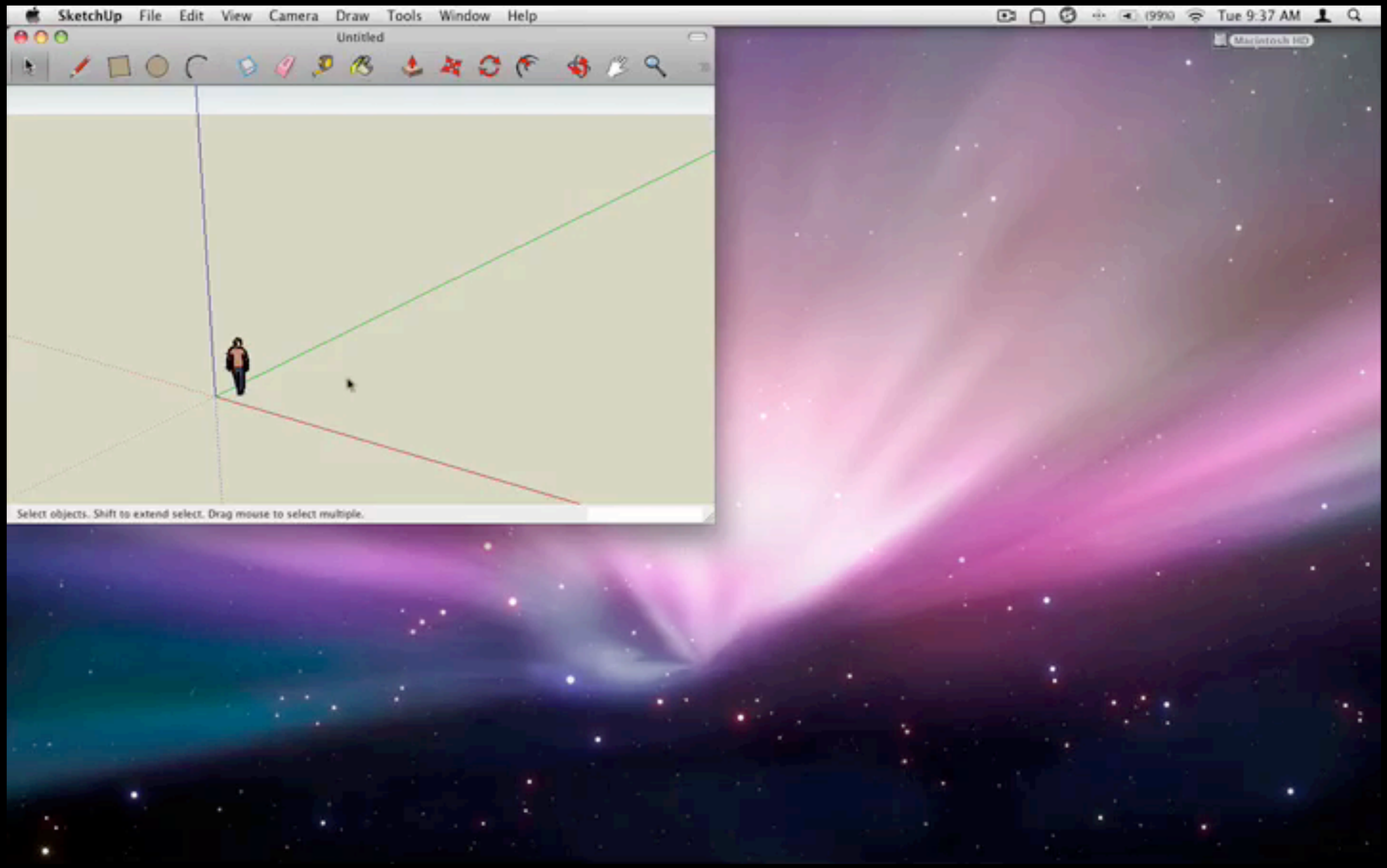
- Define motion planning problem:
  - Define an environment
  - Define a robot
  - Specify queries
  - Specify global planner (PRM, RRT, etc.)
  - Specify local planner
- Run OOPSMP to solve motion planning problem

Google SketchUp:
a graphical front-end to OOPSMP

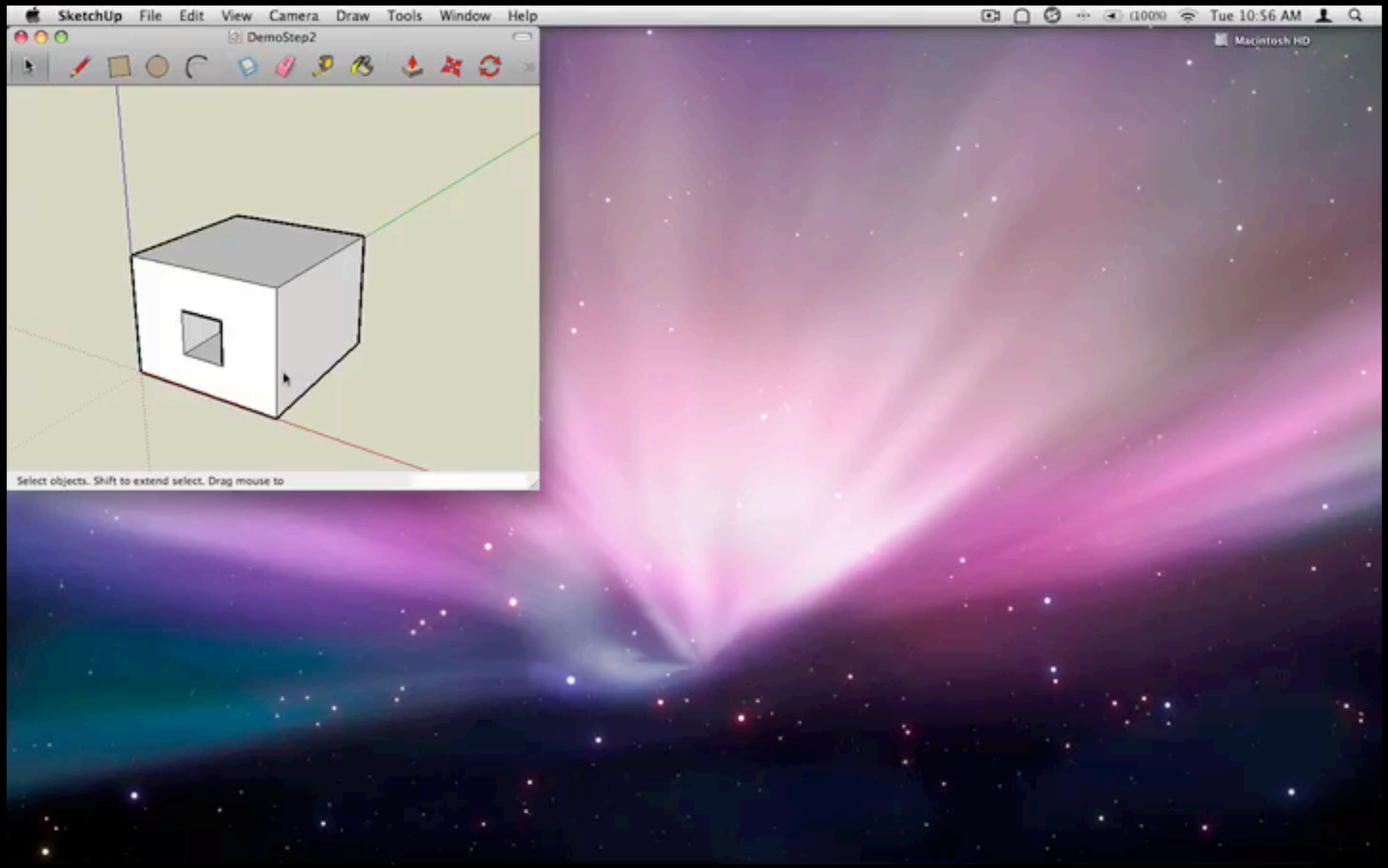
- SketchUp is a 3D modeling program
- N. Bridle & N. Feltman in our group have developed a SketchUp plugin that interfaces with OOPSMP
- Pros: ease of use, access to many models
- Cons: does not (yet) support all OOPSMP features, not available for Linux

Motion planning with OOPSMP & SketchUp

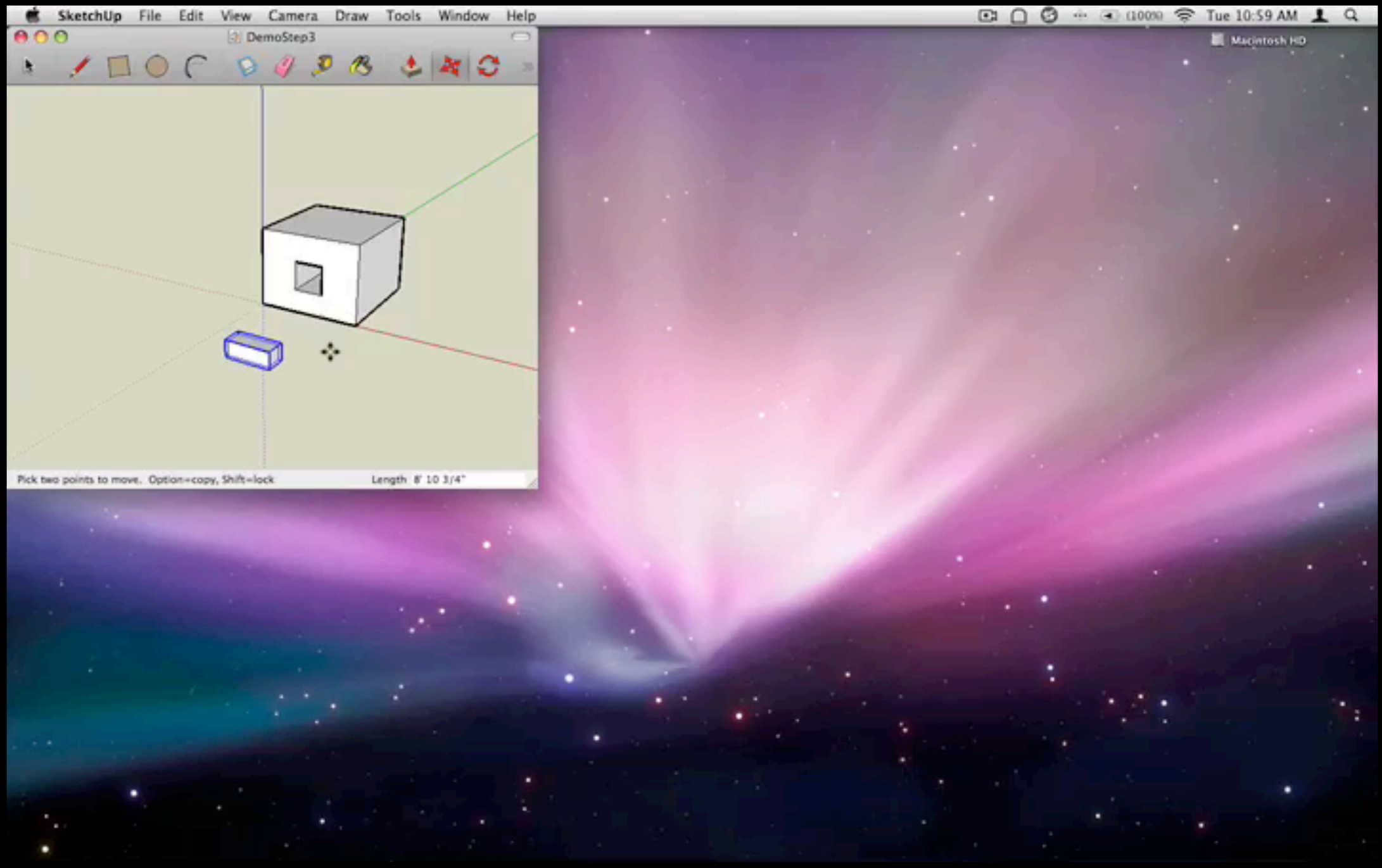
Step 1: define environment



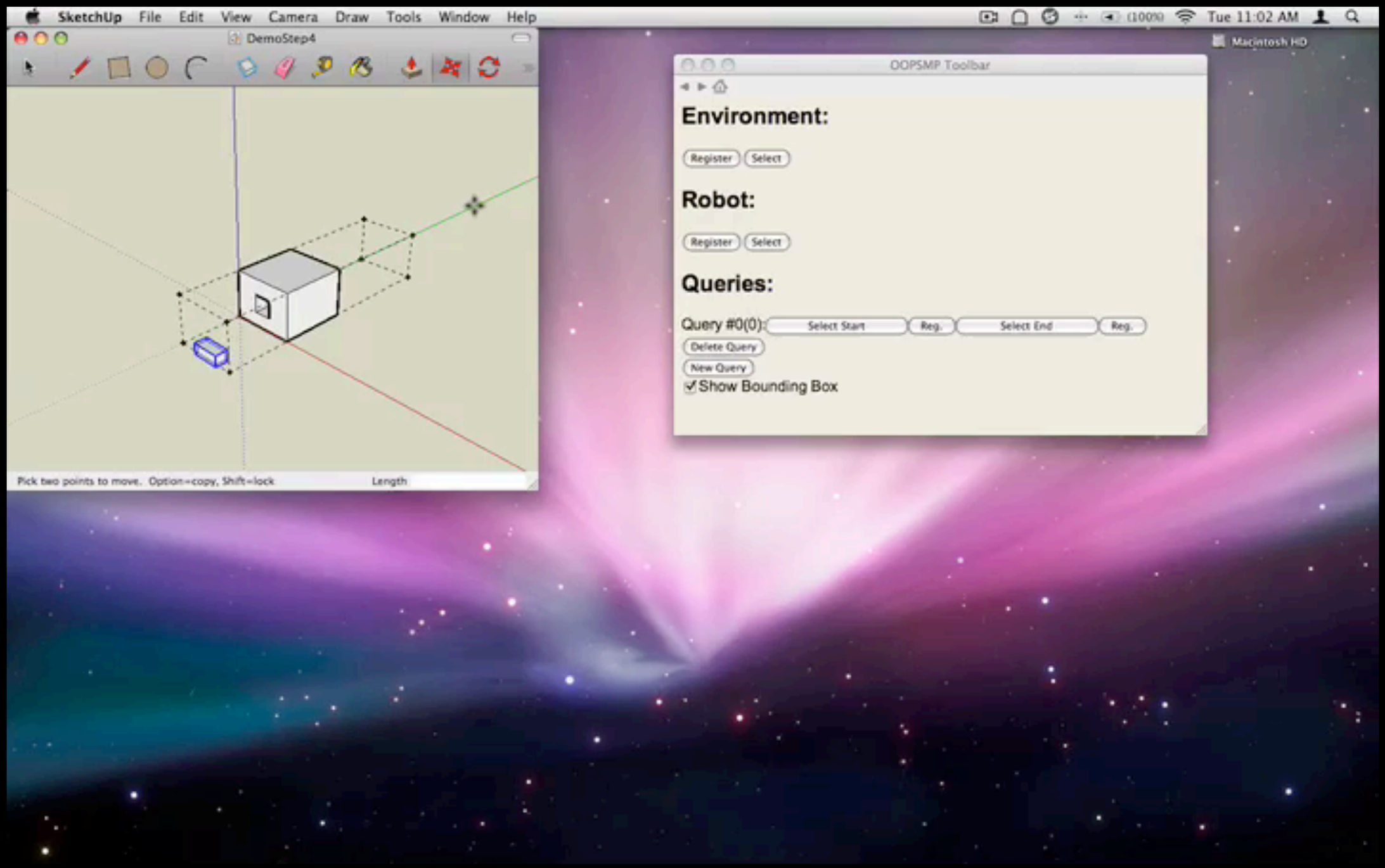
Step 2: define robot



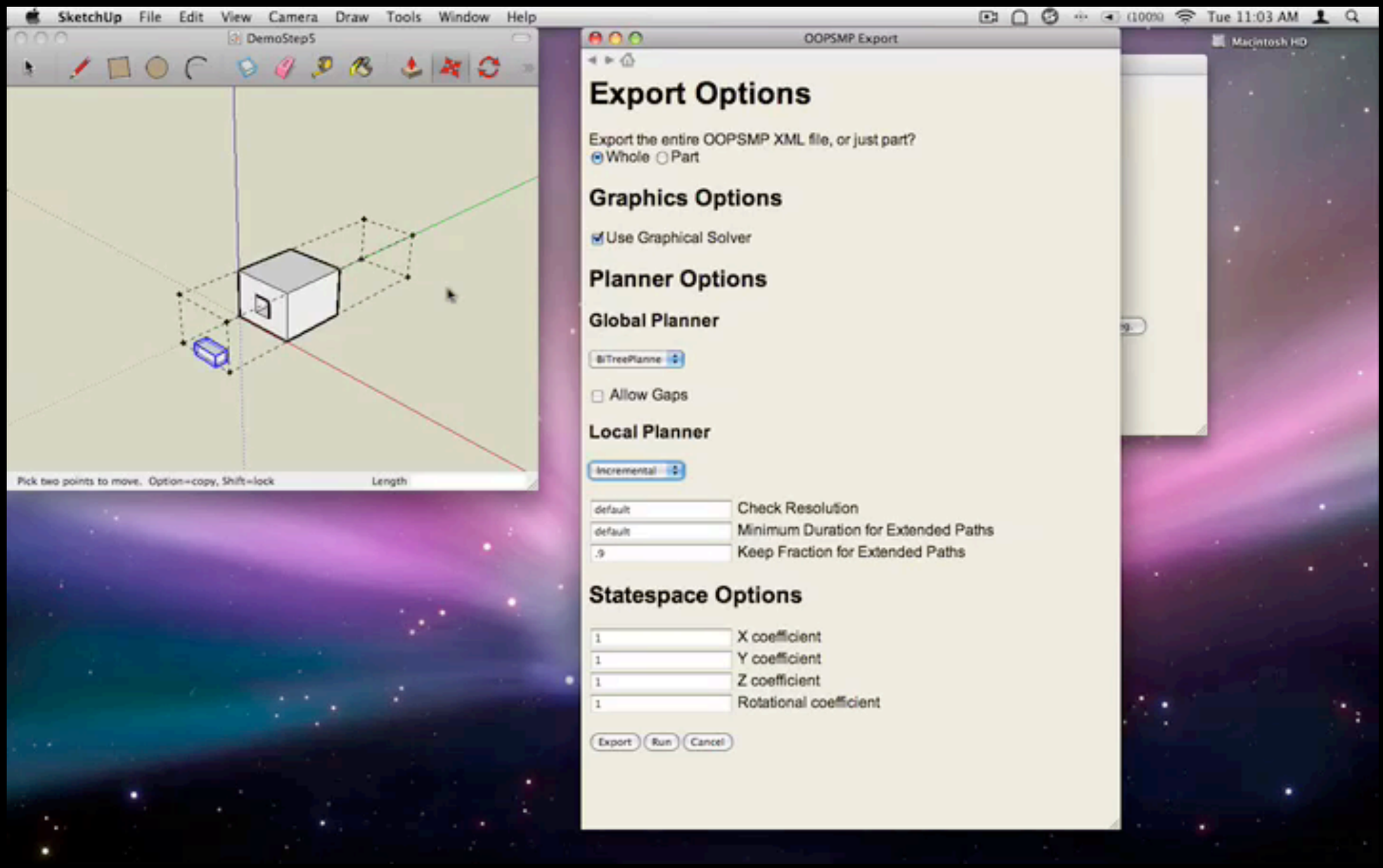
Step 3: specify queries



Step 4: specify global & local planner

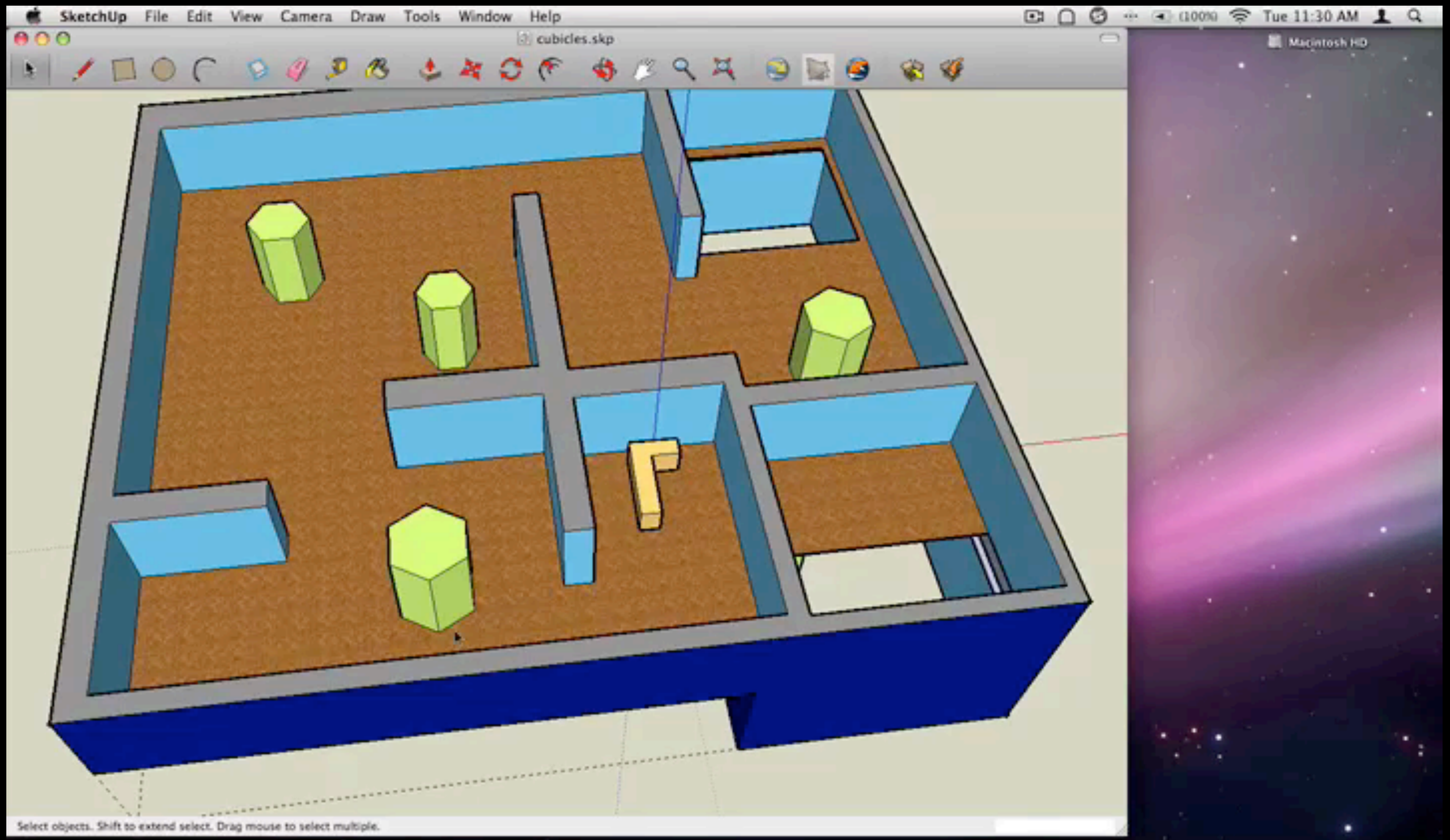


Step 5: solve queries with OOPSMP



Another example

Another example



Using OOPSMP
(without SketchUp)

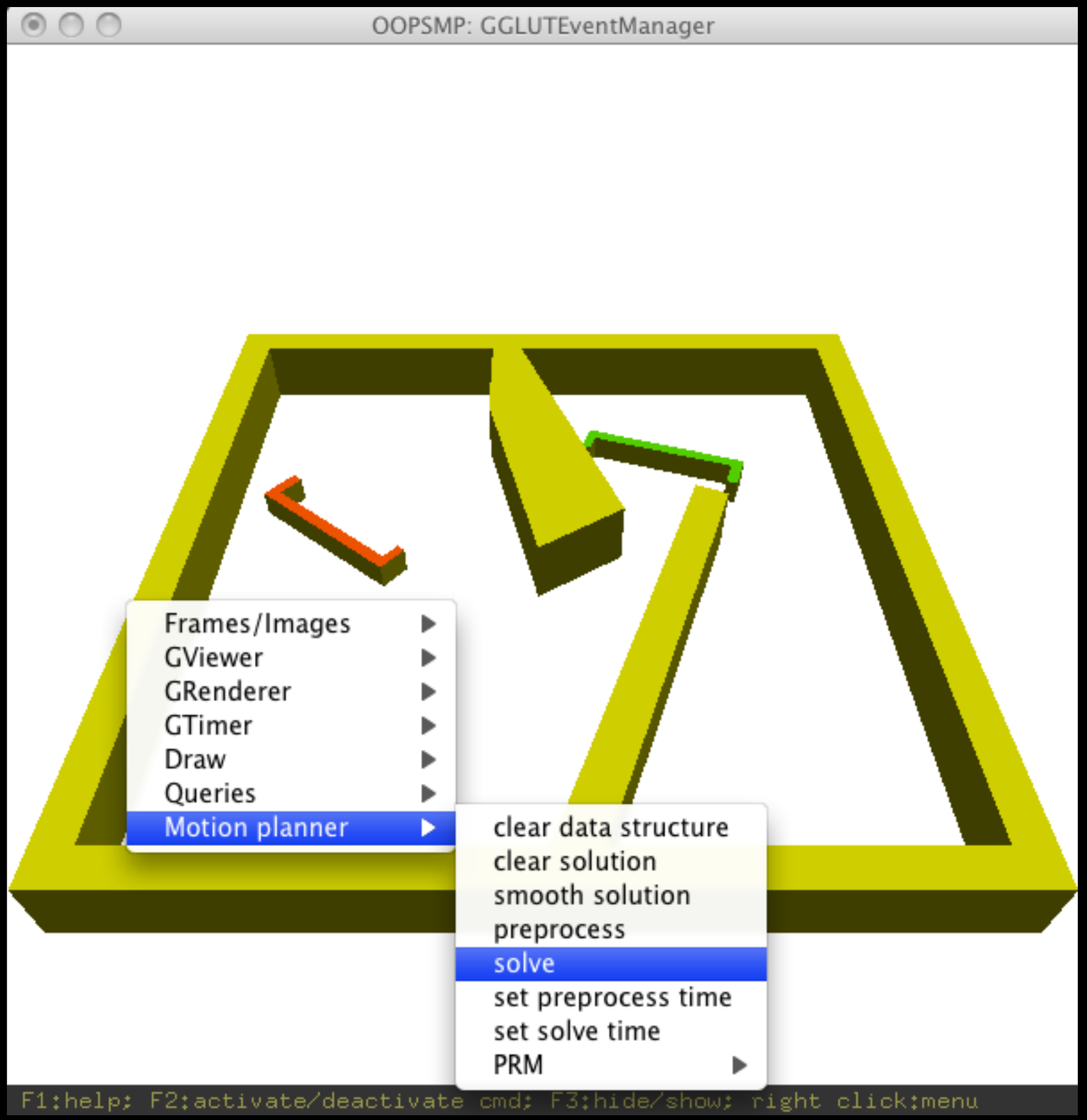
OOPSMP input

- XML input
  - Pros: self-describing, easy to parse/generate
  - Cons: very verbose
- OOPSMP XML files can include other XML files
  - Can reuse environment/robot for many motion planning problems

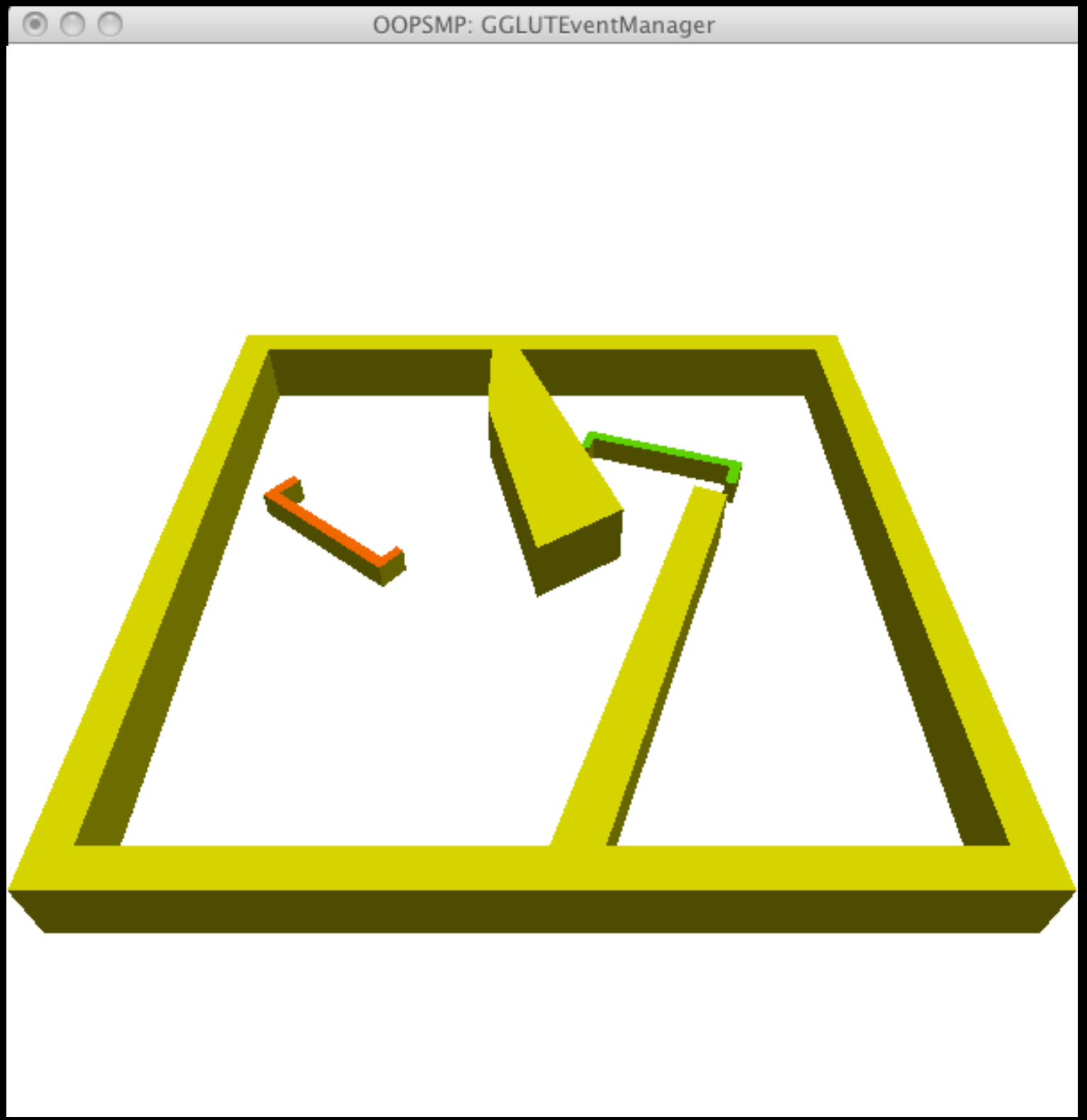
Running OOPSMP non-interactively

```
> ./bin/OOPSMPExecutable ../input/xml/DemoPrograms/DemoSE2_PRM_MotionPlannerProgr
am.xml lib/*.dylib
...OOPSMP parse xml <../input/xml/DemoPrograms/DemoSE2_PRM_MotionPlannerProgram.x
ml>
...OOPSMP inline xml
...OOPSMP load dlls
     dll file <lib/libOOPSMPCore.dylib> [ok]
     dll file <lib/libOOPSMPGCore.dylib> [ok]
     dll file <lib/libOOPSMPGManagers.dylib> [ok]
     dll file <lib/libOOPSMPGMotionPlanners.dylib> [ok]
     dll file <lib/libOOPSMPGUserPrograms.dylib> [ok]
     dll file <lib/libOOPSMPGUtils.dylib> [ok]
     dll file <lib/libOOPSMPManagers.dylib> [ok]
     dll file <lib/libOOPSMPMotionPlanners.dylib> [ok]
     dll file <lib/libOOPSMPUserPrograms.dylib> [ok]
     dll file <lib/libOOPSMPUtils.dylib> [ok]
     dll file <lib/libPQP.dylib> [ok]
     dll file <lib/libTriangulate.dylib> [ok]
...OOPSMP plugins
MotionPlanningDataFile: ../input/xml/data/MotionPlanningDataFile.txt
MotionPlanningSolutionsFile: ../input/xml/data/MotionPlanningSolutionsFile.txt
MotionPlanningQueriesFile: ../input/xml/Core/Query/Queries.xml
GraphicsFile: graphics.ps
     plugin FileNames                              ... [ok]
     plugin Queries                                ... [ok]
     plugin Part2D                                 ... [ok]
```

Running OOPSMP interactively



Running OOPSMP interactively



XML code for example

PRM.xml

```
1  <factory instance="Manager">
2    <call fn="setProgram">
3      <arg type="pointer">
4        <factory instance="MotionPlannerProgram">
5          <call fn="setFileNames">…
19         <call fn="setQueries">…
51         <call fn="setCoreRobotData">…
220        <call fn="setMotionPlanner">…
247        <call fn="setPreprocessTime">…
250        <call fn="setSolveTime">…
253      </factory>
254    </arg>
255  </call>
256</factory>
```

Line: 257 Column: 1 XML Tab Size: 2

XML code for example

PRM.xml

```
1  <factory instance="Manager">
2    <call fn="setProgram">
3      <arg type="pointer">
4        <factory instance="MotionPlannerProgram">
5          <call fn="setFileNames">…
19         <call fn="setQueries">…
51         <call fn="setCoreRobotData">…
220        <call fn="setMotionPlanner">…
247        <call fn="setPreprocessTime">…
250        <call fn="setSolveTime">…
253      </factory>
254    </arg>
255  </call>
256</factory>
```

create object

Line: 257 Column: 1 XML Tab Size: 2

XML code for example

PRM.xml

```
1  <factory instance="Manager">
2    <call fn="setProgram">
3      <arg type="pointer">
4        <factory instance="MotionPlannerProgram">
5          <call fn="setFileNames">…
19         <call fn="setQueries">…
51         <call fn="setCoreRobotData">…
220        <call fn="setMotionPlanner">…
247        <call fn="setPreprocessTime">…
250        <call fn="setSolveTime">…
253      </factory>
254    </arg>
255  </call>
256</factory>
```

create object

call object's method

Line: 257 Column: 1 XML Tab Size: 2

XML code maps to C++ objects/methods

Main Page | Namespaces | Classes | Files | Directories | Related Pages

Alphabetical List | Class List | Class Hierarchy | Class Members

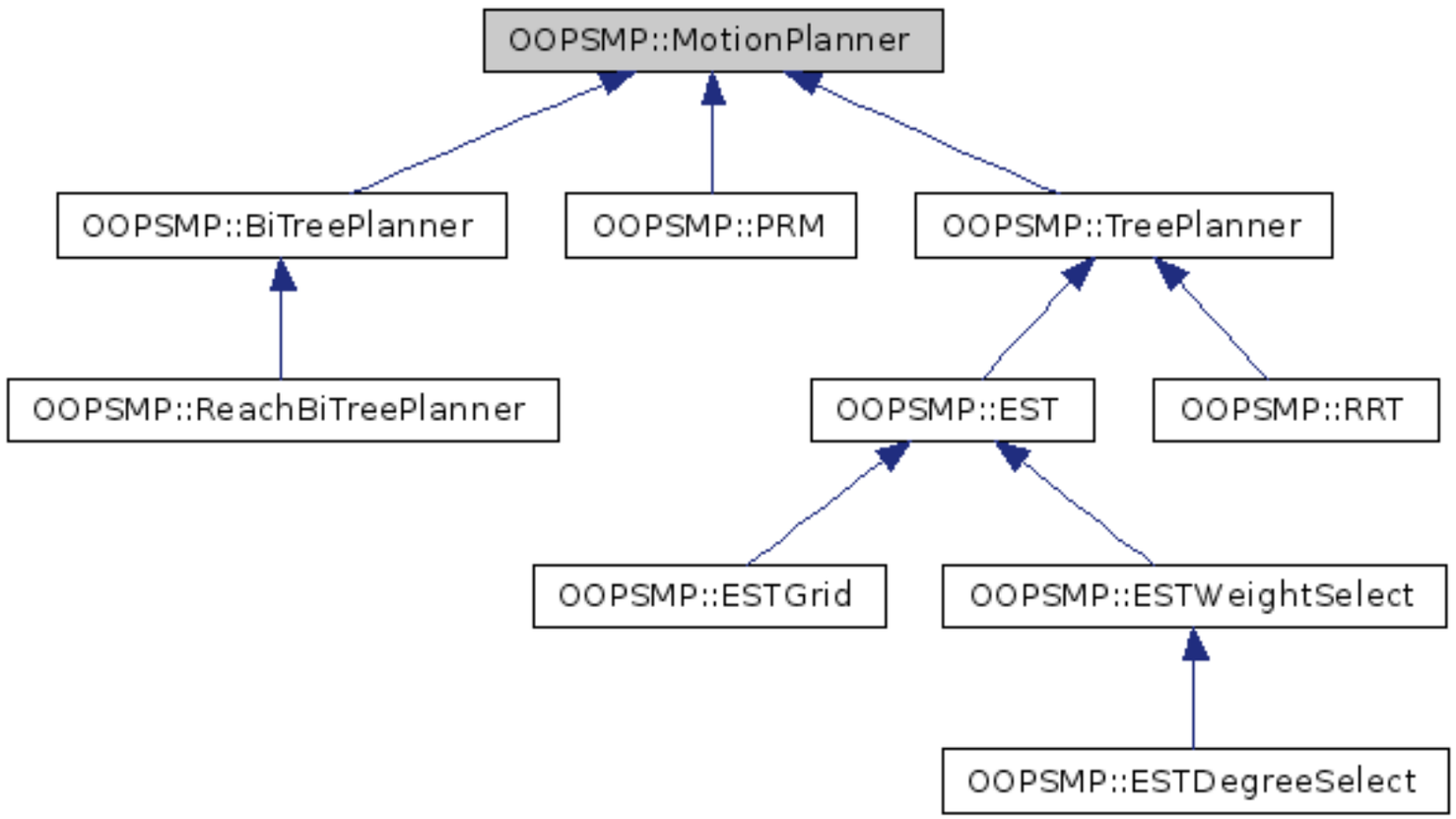
OOPSMP::MotionPlanner

OOPSMP::MotionPlanner Class Reference

Common functionality every motion planner should provide [written by Erion Plaku]. More...

#include <OOPSMPMotionPlanner.H>

Inheritance diagram for OOPSMP::MotionPlanner:



[legend]

XML code maps to C++ objects/methods

Free the memory allocated to different data elements.

Access motion planner data

bool	**areGapsAllowed** (void) const *Indicate whether gaps are allowed in path connections, i.e., path from a to b does not necessarily end at b , but somewhere near b.*
Query_t	**getQuery** (void) const *Access the current query the motion planner is solving.*
virtual bool	**isProblemSolved** (void) const=0 *Indicate whether the current motion planning query has been solved or not.*

Setup motion planner

void	**setCoreRobotData** (CoreRobotData_t crd) *Setup the core robot data, i.e., motion planning components.*
virtual void	**allowGaps** (const bool allowGaps) *Indicate whether or not gaps are allowed, i.e., path from a to b does not necessarily end at b , but somewhere near b.*
virtual void	**setQuery** (Query_t query) ***Set** the motion planning qeury.*

Use motion planner to solve one or more queries

virtual void	**solveMultipleQueries** (const double ptime, const double stime, Queries_t queries, const int *perm=NULL)=0 *Solve multiple motion planning queries.*
virtual void	**solveSingleQuery** (const double ptime, const double stime, Query_t query) *Solve a single motion planning query.*

Solve the current motion planner query

XML code for example

PRM.xml

```
1  <factory instance="Manager">
2    <call fn="setProgram">
3      <arg type="pointer">
4        <factory instance="MotionPlannerProgram">
5          <call fn="setFileNames">…
19         <call fn="setQueries">…
51         <call fn="setCoreRobotData">…
220        <call fn="setMotionPlanner">…
247        <call fn="setPreprocessTime">…
250        <call fn="setSolveTime">…
253      </factory>
254    </arg>
255  </call>
256</factory>
```

Line: 257 Column: 1 XML Tab Size: 2

Output files

PRM.xml

```
<factory instance="Manager">
  <call fn="setProgram">
    <arg type="pointer">
      <factory instance="MotionPlannerProgram">
        <call fn="setFileNames">
          <arg type="pointer">
            <factory instance="FileNames">
              <call fn="addFileName">
                <arg type="string">MotionPlanningDataFile </arg>
                <arg type="string">MyDataFile.txt</arg>
              </call>
              <call fn="addFileName">
                <arg type="string">MotionPlanningSolutionsFile</arg>
                <arg type="string">MySolutionsFile.txt</arg>
              </call>
            </factory>
```

Line: 17 Column: 17 XML Tab Size: 2

Output files

PRM.xml

```
1  <factory instance="Manager">
2    <call fn="setProgram">
3      <arg type="pointer">
4        <factory instance="MotionPlannerProgram">
5          <call fn="setFileNames">
6            <arg type="pointer">
7              <factory instance="FileNames">
8                <call fn="addFileName">
9                  <arg type="string">MotionPlanningDataFile </arg>
10                 <arg type="string">MyDataFile.txt</arg>
11               </call>
12               <call fn="addFileName">
13                 <arg type="string">MotionPlanningSolutionsFile</arg>
14                 <arg type="string">MySolutionsFile.txt</arg>
15               </call>
16             </factory>
```

roadmap

Line: 17 Column: 17 XML Tab Size: 2

Output files

PRM.xml

```
1  <factory instance="Manager">
2    <call fn="setProgram">
3      <arg type="pointer">
4        <factory instance="MotionPlannerProgram">
5          <call fn="setFileNames">
6            <arg type="pointer">
7              <factory instance="FileNames">
8                <call fn="addFileName">
9                  <arg type="string">MotionPlanningDataFile </arg>
10                 <arg type="string">MyDataFile.txt</arg>
11               </call>
12               <call fn="addFileName">
13                 <arg type="string">MotionPlanningSolutionsFile</arg>
14                 <arg type="string">MySolutionsFile.txt</arg>
15               </call>
16             </factory>
```

roadmap

solutions to queries

Line: 17 Column: 17 XML Tab Size: 2

Queries

PRM.xml

```
19        <call fn="setQueries">
20          <arg type="pointer">
21            <factory instance="Queries">
22              <call fn="addQueries">
23                <arg type="int">10</arg>
24                <arg type="int">3</arg>
25                <arg type="double_array">
26                  -0.022911 -0.115962 -0.729750
27                  -0.335974 0.286217 1.640217
28                  -0.361374 0.322786 2.294763
29                  -0.203585 -0.072764 -1.362865
30                  0.198168 0.257312 2.805693
31                  -0.324529 0.097850 -0.686631
32                  -0.057077 -0.080242 0.110432
33                  -0.173286 -0.009210 2.089135
```

Line: 25 Column: 31 XML Tab Size: 2

Queries

PRM.xml

```
19        <call fn="setQueries">
20          <arg type="pointer">
21            <factory instance="Queries">
22              <call fn="addQueries">
23                <arg type="int">10</arg>
24                 <arg type="int">3</arg>
25                <arg type="double_array">
26                  -0.022911 -0.115962 -0.729750
27                  -0.335974 0.286217 1.640217
28                  -0.361374 0.322786 2.294763
29                  -0.203585 -0.072764 -1.362865
30                  0.198168 0.257312 2.805693
31                  -0.324529 0.097850 -0.686631
32                  -0.057077 -0.080242 0.110432
33                  -0.173286 -0.009210 2.089135
```

10 queries

Line: 25 Column: 31 XML Tab Size: 2

Queries

PRM.xml

```
19        <call fn="setQueries">
20          <arg type="pointer">
21            <factory instance="Queries">
22              <call fn="addQueries">
23                <arg type="int">10</arg>
24                <arg type="int">3</arg>
25                <arg type="double_array">
26                  -0.022911 -0.115962 -0.729750
27                  -0.335974 0.286217 1.640217
28                  -0.361374 0.322786 2.294763
29                  -0.203585 -0.072764 -1.362865
30                  0.198168 0.257312 2.805693
31                  -0.324529 0.097850 -0.686631
32                  -0.057077 -0.080242 0.110432
33                  -0.173286 -0.009210 2.089135
```

10 queries

dim. of state space

Line: 25 Column: 31 XML Tab Size: 2

Robot & environment

PRM.xml

```
4    <factory instance="MotionPlannerProgram">
5      <call fn="setFileNames">…
19     <call fn="setQueries">…
51     <call fn="setCoreRobotData">
52       <arg type="pointer">
53         <factory instance="CoreRobotData">
54           <call fn="setWorkspace">…
172          <call fn="setCollisionDetector">…
177          <call fn="setStateSpace">…
192          <call fn="setValidStateSampler">…
197          <call fn="setPathGenerator">…
202          <call fn="setLocalPlanner">…
217        </factory>
218      </arg>
219    </call>
```

Line: 202 Column: 42 XML Tab Size: 2

Robot & environment

PRM.xml

```
4   <factory instance="MotionPlannerProgram">
5     <call fn="setFileNames">…
19    <call fn="setQueries">…
51    <call fn="setCoreRobotData">
52      <arg type="pointer">
53        <factory instance="CoreRobotData">
54          <call fn="setWorkspace">…
172         <call fn="setCollisionDetector">…
177         <call fn="setStateSpace">…
192         <call fn="setValidStateSampler">…
197         <call fn="setPathGenerator">…
202         <call fn="setLocalPlanner">…
217       </factory>
218     </arg>
219   </call>
```

robot and object geometries

Line: 202 Column: 42 XML Tab Size: 2

Robot & environment

PRM.xml

```
4    <factory instance="MotionPlannerProgram">
5      <call fn="setFileNames">…
19     <call fn="setQueries">…
51     <call fn="setCoreRobotData">
52       <arg type="pointer">
53         <factory instance="CoreRobotData">
54           <call fn="setWorkspace">…
172          <call fn="setCollisionDetector">…
177          <call fn="setStateSpace">…
192          <call fn="setValidStateSampler">…
197          <call fn="setPathGenerator">…
202          <call fn="setLocalPlanner">…
217        </factory>
218      </arg>
219    </call>
```

PQP 2D/3D

Line: 202 Column: 42 XML Tab Size: 2

Robot & environment

PRM.xml

```
4    <factory instance="MotionPlannerProgram">
5      <call fn="setFileNames">…
19     <call fn="setQueries">…
51     <call fn="setCoreRobotData">
52       <arg type="pointer">
53         <factory instance="CoreRobotData">
54           <call fn="setWorkspace">…
172          <call fn="setCollisionDetector">…
177          <call fn="setStateSpace">…
192          <call fn="setValidStateSampler">…
197          <call fn="setPathGenerator">…
202          <call fn="setLocalPlanner">…
217        </factory>
218      </arg>
219    </call>
```

SE(2), R^3, etc.

Line: 202 Column: 42 XML Tab Size: 2

Robot & environment

PRM.xml

```
4    <factory instance="MotionPlannerProgram">
5      <call fn="setFileNames">…
19     <call fn="setQueries">…
51     <call fn="setCoreRobotData">
52       <arg type="pointer">
53         <factory instance="CoreRobotData">
54           <call fn="setWorkspace">…
172          <call fn="setCollisionDetector">…
177          <call fn="setStateSpace">…
192          <call fn="setValidStateSampler">…
197          <call fn="setPathGenerator">…
202          <call fn="setLocalPlanner">…
217        </factory>
218      </arg>
219    </call>
```

uniform, Gaussian,
bridge, obstacle-based

Line: 202 Column: 42 XML Tab Size: 2

Robot & environment

PRM.xml

```
4    <factory instance="MotionPlannerProgram">
5      <call fn="setFileNames">…
19     <call fn="setQueries">…
51     <call fn="setCoreRobotData">
52       <arg type="pointer">
53         <factory instance="CoreRobotData">
54           <call fn="setWorkspace">…
172          <call fn="setCollisionDetector">…
177          <call fn="setStateSpace">…
192          <call fn="setValidStateSampler">…
197          <call fn="setPathGenerator">…
202          <call fn="setLocalPlanner">…
217        </factory>
218      </arg>
219    </call>
```

Line: 202 Column: 42 XML Tab Size: 2

geodesics,
controlled path, etc.

Robot & environment

PRM.xml

```
4    <factory instance="MotionPlannerProgram">
5      <call fn="setFileNames">…
19     <call fn="setQueries">…
51     <call fn="setCoreRobotData">
52       <arg type="pointer">
53         <factory instance="CoreRobotData">
54           <call fn="setWorkspace">…
172          <call fn="setCollisionDetector">…
177          <call fn="setStateSpace">…
192          <call fn="setValidStateSampler">…
197          <call fn="setPathGenerator">…
202          <call fn="setLocalPlanner">…
217        </factory>
218      </arg>
219    </call>
```

incremental or subdivision

Line: 202 Column: 42 XML Tab Size: 2

Motion planning algorithm

PRM.xml

```
220        <call fn="setMotionPlanner">
221          <arg type="pointer">
222            <factory instance="PRM">
223              <call fn="setProximity">…
228              <call fn="setNrNeighborsSample">…
231              <call fn="setNrNeighborsQuery">…
234              <call fn="allowCycles">…
237              <call fn="setFractionSamplingTime">…
240              <call fn="setMinMaxNrSamples">…
244            </factory>
245          </arg>
246        </call>
247        <call fn="setPreprocessTime">…
250        <call fn="setSolveTime">…
```

Line: 255 Column: 10 XML Tab Size: 2

Concluding remarks

With OOPSMP it is relatively easy to:

Concluding remarks

With OOPSMP it is relatively easy to:

- “play” with various state-of-the-art sampling-based motion planning algorithms,

Concluding remarks

With OOPSMP it is relatively easy to:

- “play” with various state-of-the-art sampling-based motion planning algorithms,
- add new algorithms,

Concluding remarks

With OOPSMP it is relatively easy to:

- “play” with various state-of-the-art sampling-based motion planning algorithms,
- add new algorithms,
- develop new benchmarks,

Concluding remarks

With OOPSMP it is relatively easy to:

- “play” with various state-of-the-art sampling-based motion planning algorithms,
- add new algorithms,
- develop new benchmarks,
- perform parameter sweeps,

Concluding remarks

With OOPSMP it is relatively easy to:

- “play” with various state-of-the-art sampling-based motion planning algorithms,
- add new algorithms,
- develop new benchmarks,
- perform parameter sweeps,
- compare algorithm performance,

Concluding remarks

With OOPSMP it is relatively easy to:

- “play” with various state-of-the-art sampling-based motion planning algorithms,
- add new algorithms,
- develop new benchmarks,
- perform parameter sweeps,
- compare algorithm performance,
- embed motion planning system in larger robotic system.

OOPSMP team

- Kostas Bekris
- Nick Bridle
- Drew Bryant
- Nicolas Feltman
- Nurit Haspel
- Allison Heath
- Lydia E. Kavraki
- Andrew Ladd
- Mark Moll
- Erion Plaku *lead developer*
- Amarda Shehu
- Ioan Şucan
- Konstantinos Tsianos
- **You!**

OOPSMP online

web site: http://kavrakilab.org/software/OOPSMP

mailing list: OOPSMP@rice.edu