```
30  *  CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
31  *  LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
32  *  ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
33  *  POSSIBILITY OF SUCH DAMAGE.
34  *
35  * Author: Bhaskara Marthi
36  *********************************************************************/
37  #include <navfn/navfn_ros.h>
38  #include <navfn/MakeNavPlan.h>
39  #include <boost/shared_ptr.hpp>
40  #include <costmap_2d/costmap_2d_ros.h>
41  #include <tf2_ros/transform_listener.h>
42
43  namespace cm=costmap_2d;
44  namespace rm=geometry_msgs;
45
46  using std::vector;
47  using rm::PoseStamped;
48  using std::string;
49  using cm::Costmap2D;
50  using cm::Costmap2DROS;
51
52  namespace navfn {
53
54  class NavfnWithCostmap : public NavfnROS
55  {
56  public:
57    NavfnWithCostmap(string name, Costmap2DROS* cmap);
58    bool makePlanService(MakeNavPlan::Request& req, MakeNavPlan::Response& resp);
59
60  private:
61    void poseCallback(const rm::PoseStamped::ConstPtr& goal);
62    Costmap2DROS* cmap_;
63    ros::ServiceServer make_plan_service_;
64    ros::Subscriber pose_sub_;
65  };
66
67
68  bool NavfnWithCostmap::makePlanService(MakeNavPlan::Request& req, MakeNavPlan::Response& r
69  {
70    vector<PoseStamped> path;
71
72    req.start.header.frame_id = "map";
73    req.goal.header.frame_id = "map";
74    bool success = makePlan(req.start, req.goal, path);
75    resp.plan_found = success;
76    if (success) {
77      resp.path = path;
78    }
```

```
79
80    return true;
81  }
82
83  void NavfnWithCostmap::poseCallback(const rm::PoseStamped::ConstPtr& goal) {
84    geometry_msgs::PoseStamped global_pose;
85    cmap_->getRobotPose(global_pose);
86    vector<PoseStamped> path;
87    makePlan(global_pose, *goal, path);
88  }
89
90
91  NavfnWithCostmap::NavfnWithCostmap(string name, Costmap2DROS* cmap) :
92    NavfnROS(name, cmap)
93  {
94    ros::NodeHandle private_nh("~");
95    cmap_ = cmap;
96    make_plan_service_ = private_nh.advertiseService("make_plan", &NavfnWithCostmap::makePla
97    pose_sub_ = private_nh.subscribe<rm::PoseStamped>("goal", 1, &NavfnWithCostmap::poseCall
98  }
99
100 } // namespace
101
102 int main (int argc, char** argv)
103 {
104   ros::init(argc, argv, "global_planner");
105
106   tf2_ros::Buffer buffer(ros::Duration(10));
107   tf2_ros::TransformListener tf(buffer);
108
109   costmap_2d::Costmap2DROS lcr("costmap", buffer);
110
111   navfn::NavfnWithCostmap navfn("navfn_planner", &lcr);
112
113   ros::spin();
114   return 0;
115 }
116
117
118
119
120
121
122
123
124
125
126
127
```