Aria Homework 1

Noah Harvey

November 5, 2014

Code

asn1.cpp

```
1  /*
2   * tst.cpp is part of mecharia.
3   *
4   * mecharia is free software: you can redistribute it and/or modify
5   * it under the terms of the GNU General Public License as published by
6   * the Free Software Foundation, either version 3 of the License, or
7   * (at your option) any later version.
8   *
9   * mecharia is distributed in the hope that it will be useful,
10  * but WITHOUT ANY WARRANTY; without even the implied warranty of
11  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
12  * GNU General Public License for more details.
13  *
14  * You should have received a copy of the GNU General Public License
15  * along with mecharia.  If not, see <http://www.gnu.org/licenses/>.
16  */
17
18 /**
19  * @file tst.cpp
20  *
21  * Aria test/learning file
22  *
23  * @author Dr. Matthew Marshall (mmarshall@spsu.edu)
24  * @author Noah Harvey (nharvey@spsu.edu)
25  *
26  * @copyright GNU Public License 2
27  */
28 #include <Aria.h>
29 #include "ariaTypedefs.h"
30 #include "ariaUtil.h"
31 #include "ArAction.h"
32
33 const int SQRDIST = 2000; /**< square shapre distance is 2 meters */
34
35 const double ps[4] = {0,SQRDIST,SQRDIST,0};
36 typedef struct
37 {
38         unsigned char xcnt:2;
39         unsigned char ycnt:2;
```

```
40 }cntr_t;
41
42 class ArActionGotoCust : public ArActionGotoStraight
43 {
44         public:
45                 ArActionGotoCust() : ArActionGotoStraight("goto",10000){};
46                 virtual ~ArActionGotoCust() {};
47                 virtual ArActionDesired* fire(ArActionDesired);
48 };
49
50 /** @brief custom fire action to have robot rotote before it moves linearly
51  *
52  * code copied from ArActionGotoStraight.h
53  */
54 ArActionDesired *ArActionGotoCust::fire(ArActionDesired currentDesired)
55 {
56   double angle;
57   double dist;
58   double distToGo;
59   double vel;
60
61   // if we're there we don't do anything
62   if (myState == STATE_ACHIEVED_GOAL || myState == STATE_NO_GOAL)
63     return NULL;
64
65   ArPose goal;
66   if (!myUseEncoderGoal)
67   {
68     goal = myGoal;
69     myDistTravelled += myRobot->getPose().findDistanceTo(myLastPose);
70     myLastPose = myRobot->getPose();
71   }
72   else
73   {
74     goal = myRobot->getEncoderTransform().doTransform(myEncoderGoal);
75     myDistTravelled += myRobot->getEncoderPose().findDistanceTo(myLastPose);
76     myLastPose = myRobot->getEncoderPose();
77   }
78
79   if (myJustDist)
80   {
81     distToGo = myDist - myDistTravelled;
82     dist = fabs(distToGo);
83   }
84   else
85   {
86     dist = myRobot->getPose().findDistanceTo(goal);
87   }
88
89   if (((myJustDist && distToGo <= 0) ||
90        (!myJustDist && dist < myCloseDist))
```

```
91          && ArMath::fabs(myRobot->getVel() < 5))
92      {
93        if (myPrinting)
94          ArLog::log(ArLog::Normal, "Achieved goal");
95        myState = STATE_ACHIEVED_GOAL;
96        myDesired.setVel(0);
97        myDesired.setDeltaHeading(0);
98        return &myDesired;
99      }
100
101     // see where we want to point
102     angle = myRobot->getPose().findAngleTo(goal);
103     if (myBacking)
104       angle = ArMath::subAngle(angle, 180);
105
106     myDesired.setHeading(angle);
107
108           /**
109            * added by Noah Harvey
110            * rotate towards our target before we start moving
111            */
112           if(myRobot->getRotVel() == 0)
113           {
114                   // if we're close, stop
115             if ((myJustDist && distToGo <= 0) ||
116                 (!myJustDist && dist < myCloseDist))
117             {
118               myDesired.setVel(0);
119               vel = 0;
120             }
121             else
122             {
123               vel = sqrt(dist * 200 * 2);
124               if (vel > mySpeed)
125                 vel = mySpeed;
126               if (myBacking)
127                 vel *= -1;
128               myDesired.setVel(vel);
129             }
130           }
131           else
132                   myDesired.setVel(0);
133
134     if (myPrinting)
135       ArLog::log(ArLog::Normal, "dist %.0f angle %.0f vel %.0f",
136                  dist, angle, vel);
137
138     return &myDesired;
139   }
140
141   void asn1(ArRobot* robot)
```

```
142 {
143         char i;
144         cntr_t cntr;
145         cntr.xcnt = 1;
146         cntr.ycnt = 0;
147
148         if(!robot)
149                 return;
150
151         ArLog::log(ArLog::Normal,"Starting Assigment 1");
152
153         /** set up robot actions */
154         ArActionGotoCust gotopnt;
155         gotopnt.setCloseDist(0);
156         gotopnt.setRobot(robot);
157         robot->addAction(&gotopnt,100);
158
159         /** turn on the motors */
160         robot->enableMotors();
161
162         /** set the target points in turn */
163         for(i = 0; i < 4; i++)
164         {
165                 gotopnt.setGoal(ArPose(ps[cntr.xcnt++],ps[cntr.ycnt++]),false,
                        false);
166                 while(!gotopnt.haveAchievedGoal());
167         }
168
169         robot->disableMotors();
170         ArLog::log(ArLog::Normal,"End of Assigment 1");
171
172         return;
173 }
```