# Planner

**progetto unibo.planner23** *unibo.planner23*

La parte proattiva di un robot consiste spesso nella esecuzione di una sequenza di movimenti che portano il robot in un posizione voluta dello spazio di lavoro.

La sequenza di movimenti può venire 'cablata' nel codice di controllo del robot o può essere costruita dinamicamente, per tenere conto dello stato del robot e dell'ambiente circostante.

Pensiamo al caso del *RobotCleaner* che deve interrompere la sua parte proattiva per eseguire un comando diverso dal un semplice : **stop/resume**, quale, ad esempio, un comando di **BacktoToHome**.

Un modo per costruire dinamicamente una sequenza di mosse con cui il robot può muoversi dalla posizione corrente **r** a un'altra posizione (libera) sulla mappa, consiste nell'utilizzo di un pianificatore (si veda *Planning*) come quello fornito in **unibo.planner23-1.0.jar**.

## Uso del planner

Il robot (virtuale o reale) viene considerato un oggetto inscrivibile in un cerchio di diametro **D**.

Muovere il robot con mossa **step(T)** con tempo **T** tale da spostare il robot (con velocità prefissata) di uno spazio **D**, permette di costruire una mappa della stanza formata da celle quadrate **DxD**. Ad esempio:

```
   0  1  2  3  4  5  6  7  x
0 |r, 1, 1, 1, 1, 1, 1,
1 |1, 1, 1, 1, X, X, 1,
2 |1, 1, 1, 1, X, X, 1,
3 |1, 1, X, 1, 1, 1, 1,
4 |1, 1, 1, 1, 1, 1, 1,
5  |X, X, X, X, X, X, X,
y
```

- **0** denota una cella mai percorsa
- **1** denota una cella librera
- **X** denota una cella occupata da un ostacolo
- **r** denota la posizione corrente del robot

## Planner23Util

Il progetto *unibo.planner23*, fornisce la classe *Planner23Util* che realizza le operazioni-base descritte in *Planning* e varie altre utlità.

| | |
|---|---|
| Codice della mappa | *RoomMap* |

### Plan in forma verbosa/compatta

La sequenza di mosse che definisce un piano può essere espressa in **forma verbosa** (ad esempio **[w, w, l, w, w]**) oppure in **forma compatta** (ad esempio **wwlww**).

## Operazioni di pianificazione

Tra le operazioni offerte dal *Planner* del progetto *unibo.planner23* vi sono le seguenti:

| | |
|---|---|
| **void setGoal( Integer x, Integer y)** | Fissa le coordinate (in unità robotiche) della posizione (cella) da raggiungere (*target*). |
| **List<Action> doPlan()** | Restituisce una sequenza di mosse (*plan*) per muovere il robot dalla sua posizione corrente al *target*. La rappresentazione in forma di **String** di questa sequenza produce la *forma verbosa* del plan). |
| **String doPlanCompact()** | Restituisce una **String** che rappresenta, in *forma compatta*, il plan da seguire per raggiungere il *target*. |
| **String planCompacted(String Plan)** | Restituisce la *forma compatta* della rappresentazione testuale verbosa di un plan. |
| **void doPathOnMap(String planrep)** | Muove il robot nella mappa, in accordo alla rappresentazione testuale planrep (in forma *verbosa* o *compatta*). |

## Planner23Util: tutte le operazioni

```
void initAI()
void setRobotState(String xs, String ys, String d)
void void setGoal( Integer x, Integer y)
```

```
List<Action> doPlan() throws Exception
String planCompacted(String Plan)
String doPlanCompact() throws Exception
List<Action> planForGoal(String x,String y) throws Exception
List<Action> planForNextDirty( ) throws Exception
String planForNextDirtyCompact( ) throws Exception
void showCurrentRobotState()
doPathOnMap(String planrep)
---------------------------------
Pair<Integer,Integer> get_curPos()
Integer getPosX()
Integer getPosY()
RobotState.Direction getDir()
String getDirection()
String setTheDirection(String dir) //dir=up|left|right|down
boolean atHome()
 boolean atPos(  Integer x, Integer y )
String robotPosInfo()
String robotOnMap()

----------------------------------
Integer getMapDimX( )
Integer getMapDimY( )
boolean mapIsEmpty()
void showMap()
String getMap()
String getMapOneLine()
Pair<Integer,Integer> getMapDims()
void loadRoomMap( String fname )
void saveRoomMap(  String fname  ) throws IOException

void moveRobotInTheMap()
void doMove(String move)  //move=w|s|a|l|d|r|rightDir|leftDir|upDir|downDir
void  setPositionOnMap( )
void updateMap( String move , String msg )
void  updateMapObstacleOnCurrentDirection( )

void setObstacleUp()
void setObstacleDown()
void setObstacleLeft()
 void setObstacleRight()
void setObstacleWall( RobotState.Direction dir, Integer x , Integer y )
void setWallDown( int dimMapx, int y  )
void setWallRight( int dimMapy, int x)
void wallFound()
```

## MainPlannerdemo.java

Il programma *MainPlannerdemo.java* cotrustisce una 'mappa puramente mentale' di una stanza senza ostacoli fissi e 'muove mentalmente' il robot nella posizione **(2,3)**.

# Robot che crea una mappa con ostacoli

**progetto planusage24** *planusage24*

## mapobstaclesvrqak.qak

Il modello *mapobstaclesvrqak.qak* ha come scopo la costruzione di una mappa della stanza con ostacoli fissi. Esso utilizza:

- *Vrqak24* per comunicare con il virtual robot
- *Planner23Util* per agiornare la mappa

Il modello parte dalla assunzione che il **bordo superiore della stanza sia privo di ostacoli** e procede esplorando lungo linee verticali, procedendo da *wallUp* a *wallDown*.

Quando incontra un ostacolo, il robot si gira di **180°** e torna nella posizione iniziale HOME, per prosguire poi ad esplorare la colonna più a destra.

Il procedimento termina quando il robot non è più in grado di esplorare una colonna a destra.

## mapobstaclesrobot.qak

Il modello *mapobstaclesrobot.qak* ha come scopo la ostruzione di una mappa della stanza con ostacoli fissi. Esso utilizza:

- *BasicRobot24* per comunicare con il virtual robot
- *Planner23Util* per la impostazione di una strategia di movimento del robot che mira ad eseguire piani (sequenze di mosse) il cui goal è raggiungere una cella al di fuori della stanza.

Quando incontra un ostacolo, il robot esegue un piano per tornare nella posizione iniziale HOME.

Poichè un tale piano deve esistere (in base alla mappa costruita fino a quel momento) e deve essere eseguibile con successo (in quanto non vi sono ostacoli mobili),

- la fase di ritorno in HOME è sfruttata per riallineare lo stato effettivo del robot quello previsto dalla mappa.

Il procedimento termina quando il planner non è più in grado di trovare un piano per raggiungere una cella al di fuori della stanza. La mappa costruita fino a quel momento potrebbe contenere anora celle non esplorate, cha vanno esplorate in una fase successiva.

**TODO: visualizzare la mappa** : fare in modo che il valore corrente della mappa sia osservabile su un dispositivo di output o, preferibilmente, su una pagina web.