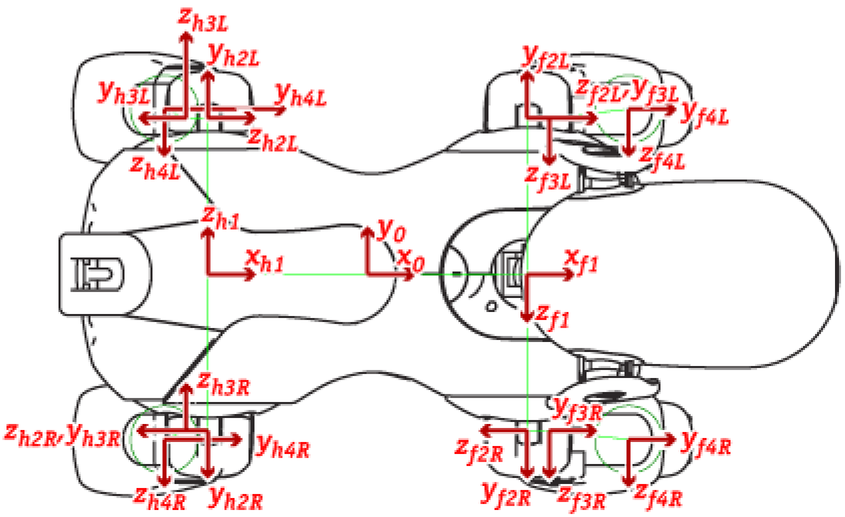
Navigating with the Pilot

15-494 Cognitive Robotics
David S. Touretzky &
Ethan Tira-Thompson

Carnegie Mellon
Spring 2010

How Does the Robot Walk?

- Multiple walk engines incorporated into Tekkotsu:
  - CMPack '02 AIBO walk engine from Veloso et al. (CMU), with modifications by Ethan Tira-Thompson
  - UPennalizers AIBO walk engine from Lee et al. (U. Penn)
  - XWalk engine by Ethan Tira-Thompson for the Chiara
- Basic idea is the same:
  - Cyclic pattern of leg motions
  - Parameters control leg trajectory, body angle, etc.
  - Many different gaits are possible by varying phases of the legs
  - “Open loop” control: no force feedback
  - Can't adapt to rough terrain
  - Can move quickly, but not very accurately

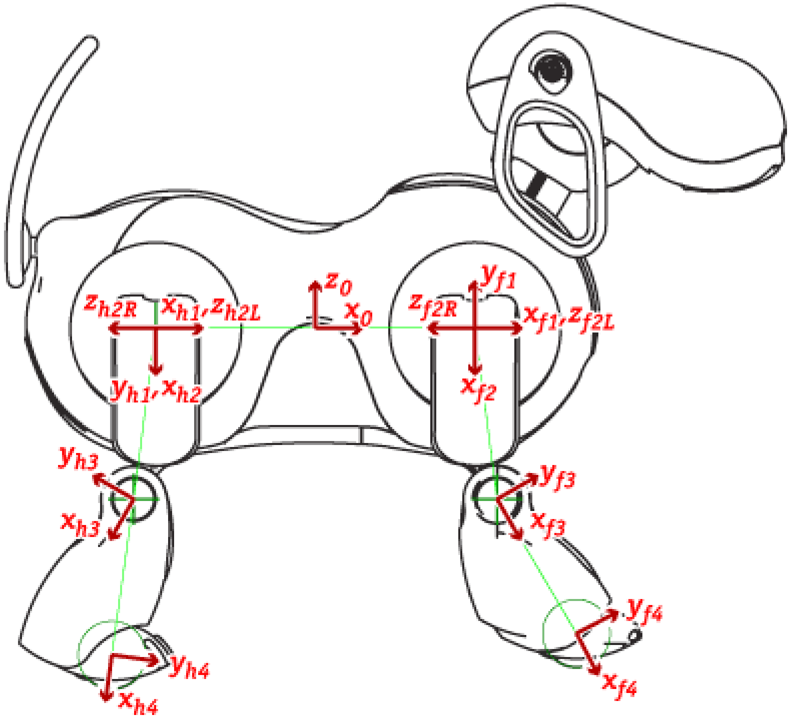


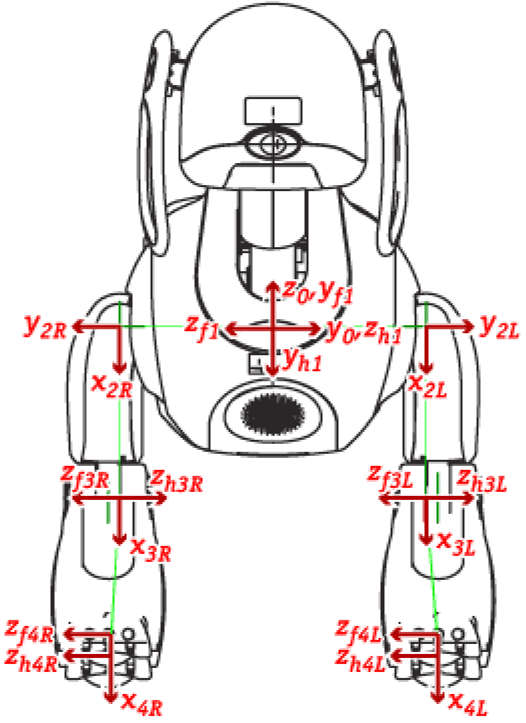
ERS-7 Legs

	Δx	Δy	Δz
1. - shoulder	65	0	0
2. - elevator	0	0	62.5
3. - knee	69.5	0	9
f4. - ball	*69.987*	*-4.993*	4.7
h4. - ball	*67.681*	*-18.503*	4.7

Diameter of ball of foot is *23.433*mm
Each link offset is relative to previous link

The shins shown in this diagram appear to be slightly distorted compared to a real robot. Corresponding measurements have been taken from actual models.





Modified CMPack Walk Engine

46 Leg Parameters:

- Neutral kinematic position (3x4)
- Lift velocity (3x4)
- Lift time (1x4)
- Down velocity (3x4)
- Down time (1x4)
- Sag distance (1)
- Differential drive (1)

5 Body Parameters:

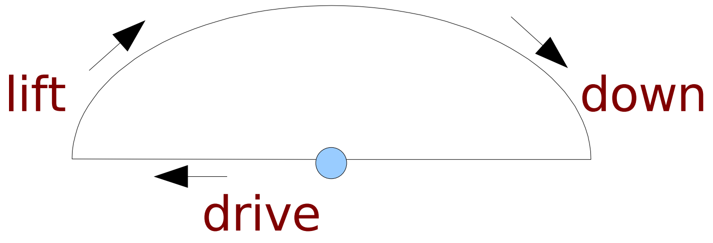
- Height of body (1)
- Angle of body (1)
- Hop amplitude (1)
- Sway amplitude (1)
- Walk period (1)

Modified fom Sonia Chernova's lecture notes

Neutral Kinematic Position

- Position (x,y,z) of the leg on the ground at some fixed point during the walk cycle.
- Where the legs would hit the ground if the robot were pacing in place (traveling with zero velocity).

Path of the leg during one walk cycle



From Sonia Chernova's lecture notes

Leg Lift and Leg Plant

- Lift velocity vector (mm/sec) determines how leg is lifted off the ground
- Down velocity vector (mm/sec) determines how leg is placed back on the ground.
- Lift time and down time (1 value each per leg) control the order of leg motions.
  - Expressed as a percentage of time through the walk cycle that the leg is raised and lowered.
  - Governs which legs move together and which move at opposite times: pace vs. trot vs. gallop.

From Sonia Chernova's lecture notes

Body Angle/Height; Hop & Sway

- Body angle (radians) relative to the ground, measured at the origin of the motion coordinate frame.
  - Controls whether the robot is pitched up or down.
- Body height (mm) relative to the ground, measured at the origin of the motion coordinate frame.
- Hop and sway amplitudes (mm) constrain the body's vertical and horizontal oscillations during walking. (Usually set to 0.)

From Sonia Chernova's lecture notes

Walk Period

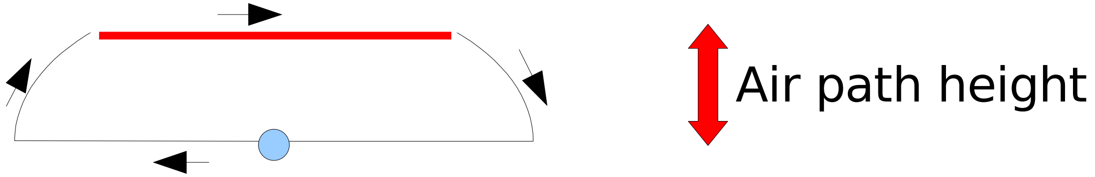
- The walk period (msec) specifies the time of one walk cycle.
- Note that this is independent of speed.
- To walk faster, the AIBO takes larger steps; it does not change the period of the walk cycle the way a person would do.

From Sonia Chernova's lecture notes

- Chiara walks are statically stable, and period does vary with speed.

New CMPack Parameter: Front & Back Leg Height Limits

- Height of the air path of the front and back legs.
- Upper bound: may not be reached, depending on other leg motion parameters.



Air path height

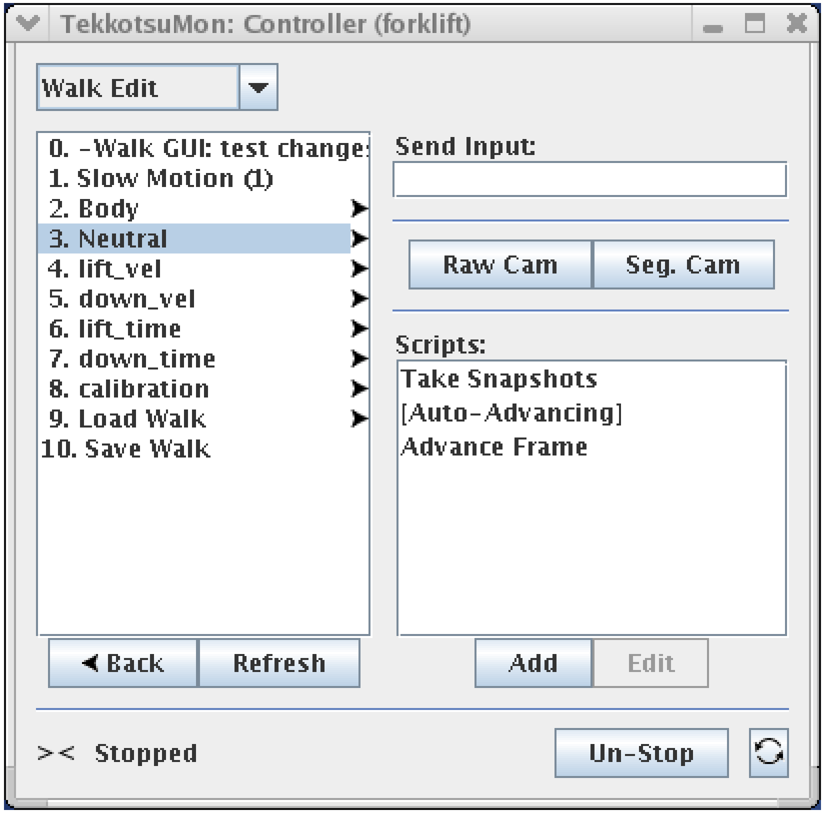
From Sonia Chernova's lecture notes

Walk Parameter Optimization

- Many RoboCup groups use machine learning techniques to optimize walk parameters.
- CMPack uses a genetic algorithm.
- Candidates are evaluated by having the robot walk and measuring the results.
- CMPack got 20% speedup over previous hand-tuned gaits.

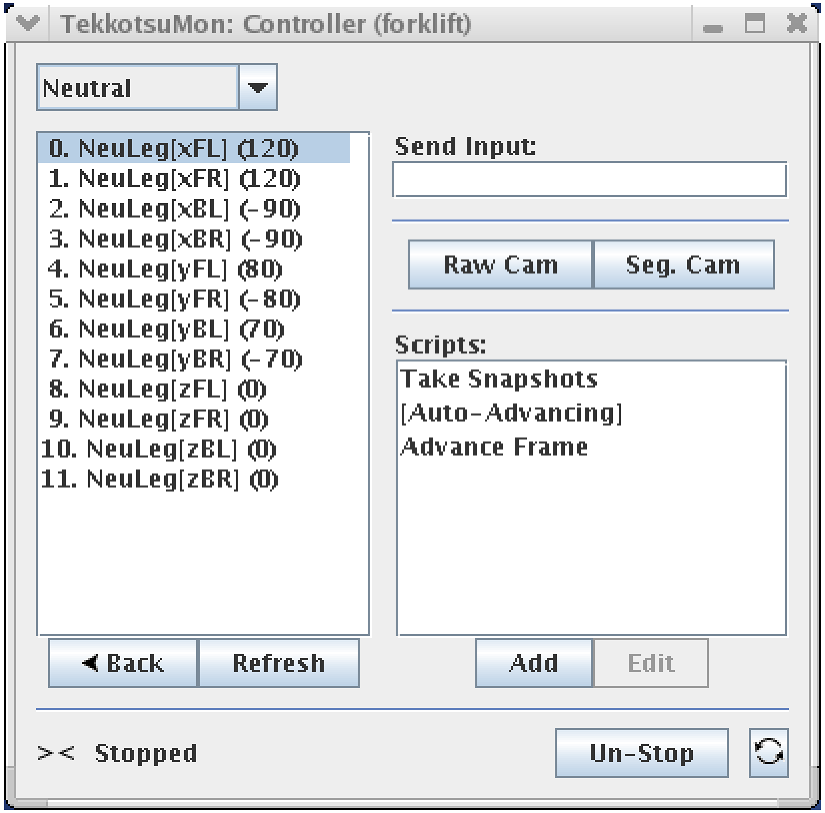
Tekkotsu Walk Editor

- Root Control > File Access > XWalk Edit
- Values are stored in a walk parameter file
  - Default parameter file is walk.plist



Walk Edit

0. -Walk GUI: test change:
1. Slow Motion (1)
2. Body
3. Neutral
4. lift_vel
5. down_vel
6. lift_time
7. down_time
8. calibration
9. Load Walk
10. Save Walk



Neutral

0. NeuLeg[xFL] (120)
1. NeuLeg[xFR] (120)
2. NeuLeg[xBL] (-90)
3. NeuLeg[xBR] (-90)
4. NeuLeg[yFL] (80)
5. NeuLeg[yFR] (-80)
6. NeuLeg[yBL] (70)
7. NeuLeg[yBR] (-70)
8. NeuLeg[zFL] (0)
9. NeuLeg[zFR] (0)
10. NeuLeg[zBL] (0)
11. NeuLeg[zBR] (0)

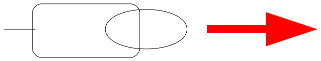
Chiara Gaits

- One leg at a time (default). walk.plist
  - Requires the least power.
  - Slow: 6 beats/cycle.
- Two legs at a time. walk2.plist
  - Intermediate speed and power.
  - 3 beats/cycle.
- Three legs at a time: tripod gait. walk3.plist
  - Fastest gait that is still statically stable.
  - Requires lots of power.
  - 2 beats/cycle.

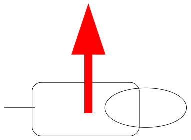
XWalkMC

- XWalkMC is a motion command that uses the Chiara walk engine to calculate leg trajectories.

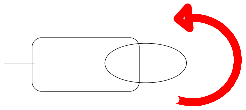
- Walking is controlled by three parameters:
  - x velocity (forward motion)



  - y velocity (lateral motion: strafing)



  - angular velocity (rotation)



XWalkNode

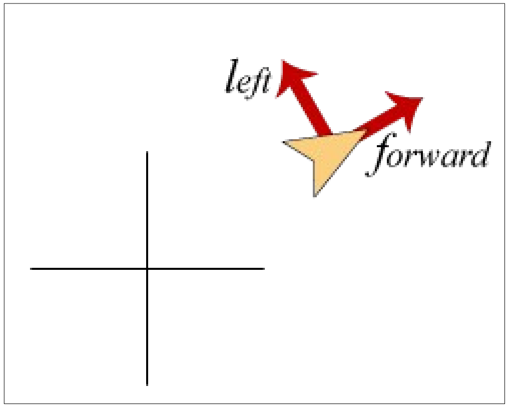
- Subclass of StateNode
- Activates an XWalkMC on start()
- Deactivates it on stop()
- Provides functions to set (x,y,a) velocities
- XWalkNode($, xvel, yvel, avel)
  - xvel, yvel in mm/sec; avel in rad/sec
- XWalkNode($, xvel, xdisp, yvel, ydisp, avel, adisp)
  - velocities in mm/sec and rad/sec; 0 means “max speed”
  - displacements in mm and rad

WalkNode

- For the iRobot Create, “walking” means driving.
- WalkNode($, xvel, yvel, avel)
  - xvel = velocity in mm/sec
  - yvel must be zero
  - avel = angular velocity in radians/s
- WalkNode($, xdist, ydist, adist, 1)
  - xdist = distance in mm
  - ydist must be zero
  - adist = angle in radians

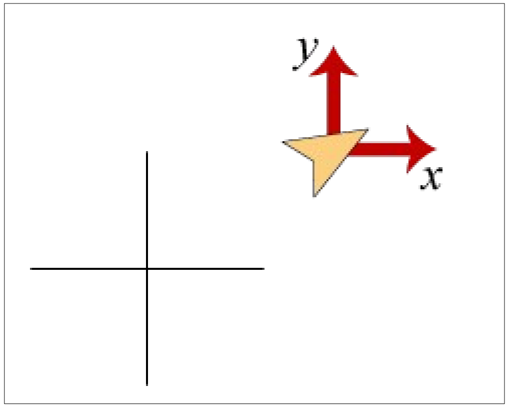
Waypoint Engine

- Takes the robot through a path defined by a series of waypoints.
- Each waypoint specifies a position (x,y) and orientation.
- Three waypoint types:



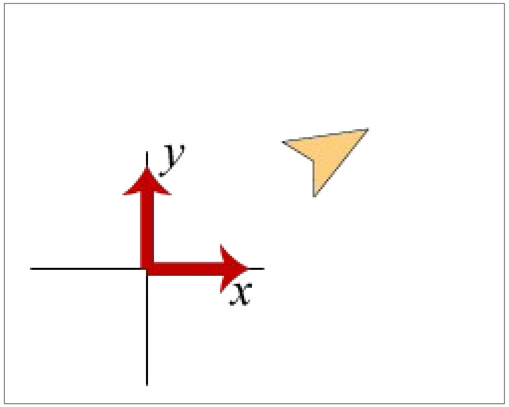
Egocentric

“Three steps forward”



Offset

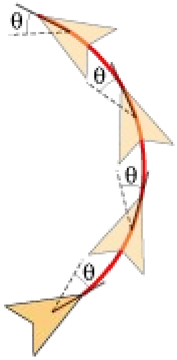
“Three steps north”



Absolute

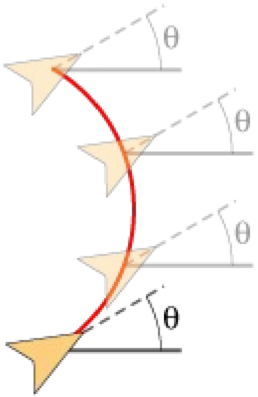
“To (30,12)”

Controlling Body Orientation



`angleIsRelative == true`

The angle is relative to the path, so an angle of 0 means the robot's body will **follow** the direction of travel.

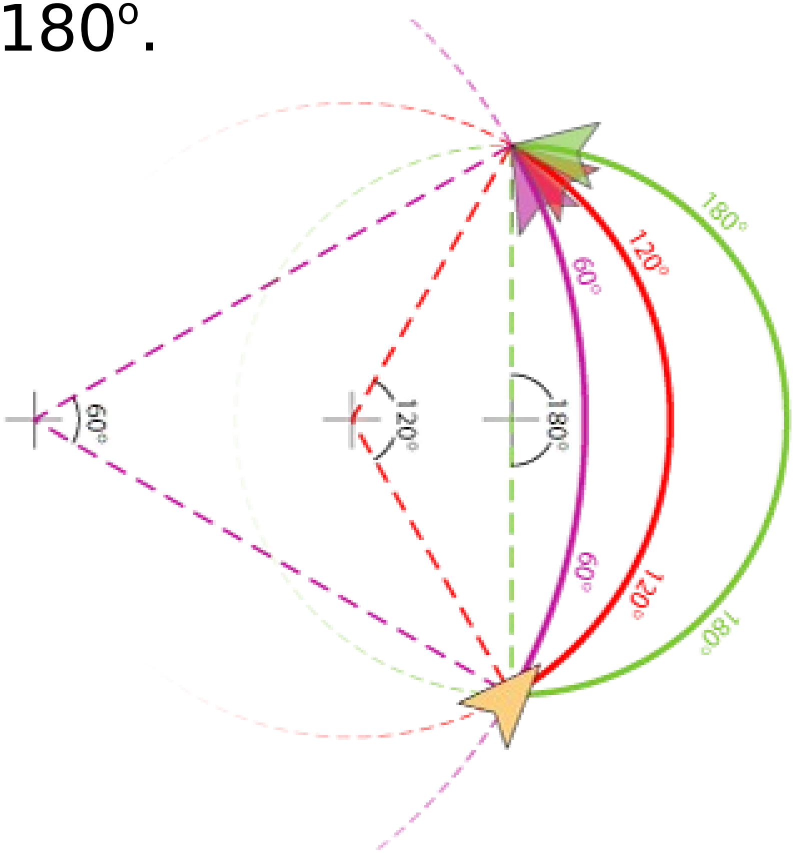


`angleIsRelative == false`

The angle is relative to the world coordinate system, so the body will **hold** a constant heading while walking.

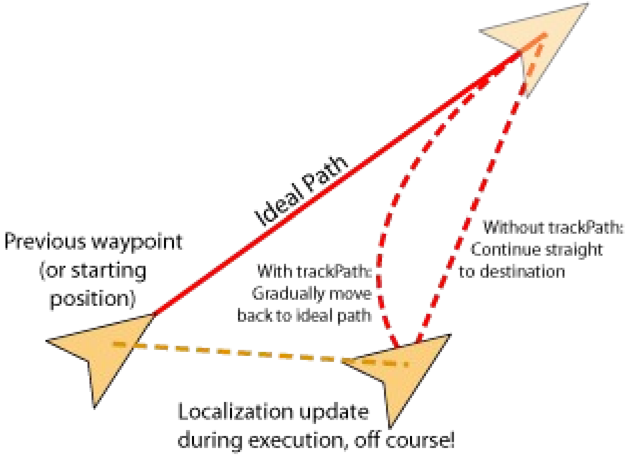
Arcing Trajectories

- Paths can be either straight lines or arcs.
- Arc parameter (in radians, not degrees) corresponds to the angle of the circle which is swept.
- Don't use values > 180°.



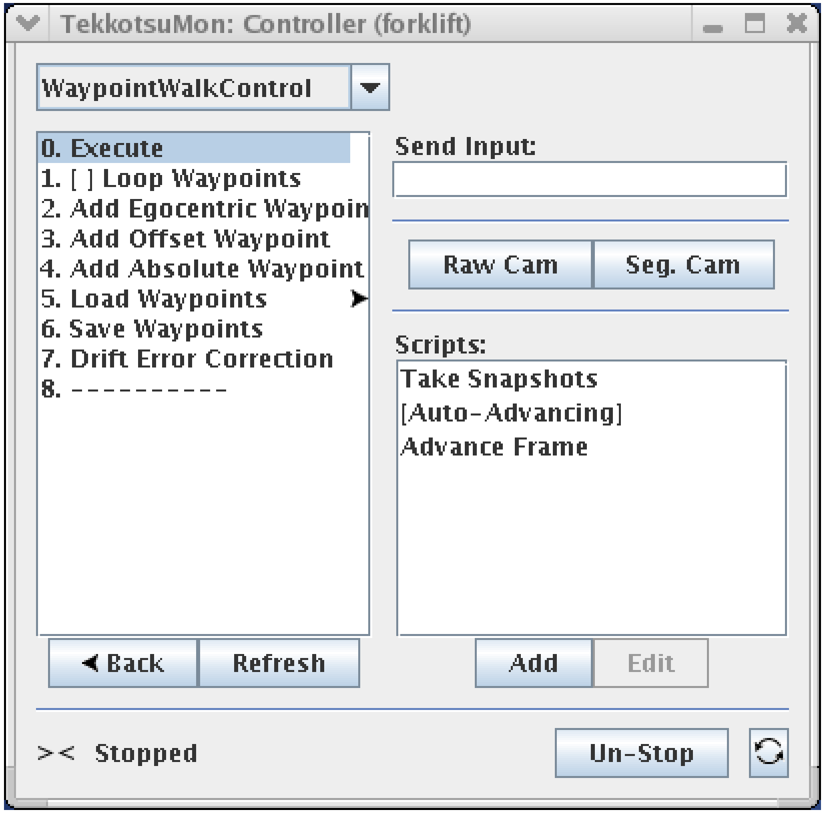
Track Path (Error Correction)

- setCurPos() function can be used to correct position if you have a localization module.
- When trackPath flag is true, the robot will attempt to return to its planned path after a perturbation.
- When false, it just goes straight to the destination.


Ideal Path
Previous waypoint (or starting position)
With trackPath: Gradually move back to ideal path
Without trackPath: Continue straight to destination
Localization update during execution, off course!

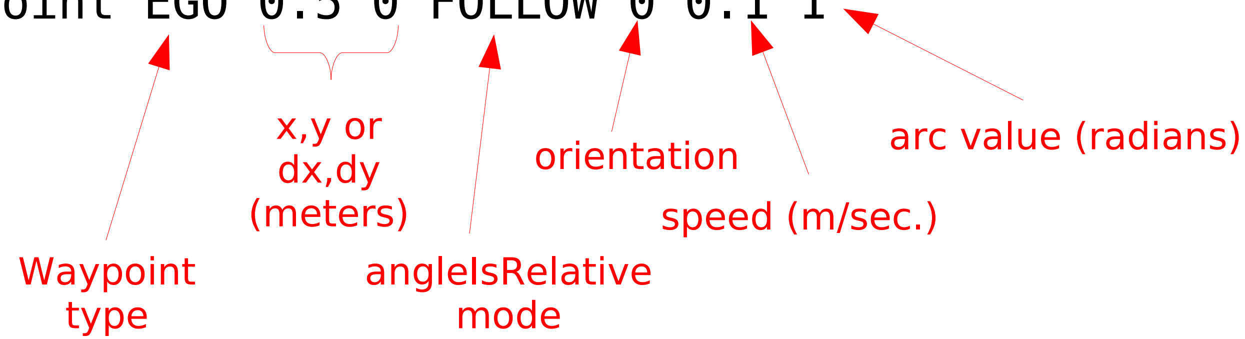
Waypoint Walk Editor

- Root Control > File Access > WaypointWalk Control
- Allows interactive creation, execution of waypoint file.



Sample Waypoint File

```
#WyP
#add_{point|arc} {ego|off|abs} x_val y_val {hold|follow} angle_val
#                                                  speed_val arc_val
max_turn_speed 0.65
track_path 0
add_point EGO 0.3 0 FOLLOW 0 0.1 0
add_point EGO 0.5 0 FOLLOW 0 0.1 1
#END
```



x,y or
dx,dy
(meters)

orientation

arc value (radians)

speed (m/sec.)

Waypoint
type

angleIsRelative
mode

WaypointWalk

- WaypointWalk is a motion command.
- Can load waypoints from a waypoint file, or construct them dynamically with function calls.
- Uses a XWalkMC to do the actual walking.
- XWalkMC will post status events indicating the progress of the walk.

The Pilot

- Higher level approach to locomotion.
- Specify effect to achieve, rather than mechanism:
  - Go to an object.
  - Maintain a bearing or distance relative to an object.
- Specify policies to use:
  - Cliff detection (IR sensor)
  - Obstacle avoidance (turn off to knock down soda cans)
  - Localization procedure
- Experimental code; changing rapidly.

Pilot Request Types

- walk
  - Essentially an XWalkMC request
- WaypointWalk
  - Waypoint walk functionality plus extras
- visualSearch
  - Use Lookout to search for an object; may rotate the body
- gotoShape
  - Travel to the location of a shape on the world map
- *More functions are planned...*

Trivial Pilot Example

```
#nodeclass MyDemo : VisualRoutinesStateNode

  #shortnodeclass Goer : PilotNode(PilotRequest::walk) : DoStart
    pilotreq.x = 500;   // forward half a meter



  #nodemethod setup

    #statemachine
      startnode: Goer =PILOT=> SpeechNode($,”I have arrived”)
    #endstatemachine

#endnodeclass
```

Manipulation by Walking

- Course project by Ethan Tira-Thompson

  http://ethan.tira-thompson.com/stuff/16-741/project.html

- Inspired by Matt Mason's “mobipulator” project.