

# Postures and Motion Sequences

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

Carnegie Mellon  
Spring 2010

# How to Move the Body

1. Set joint angles directly with a Motion Command.
  - HeadPointerMC::setJoints(...)
2. Specify a desired effect with a Motion Command.
  - HeadPointerMC::lookAtPoint(...), or WalkMC
3. Load a pre-defined posture from a posture file.
4. Play a pre-defined motion sequence from a .mot file.
5. Solve inverse kinematics problems for effector positions.
6. “Kinesthetic intelligence”: reasoning about balance, friction, joint loads, etc.

# What is a “Posture”?

- A set of zero or more effector settings:
  - effector name (e.g., LFr:knee, or LED:blue)
  - effector value (joint angle; LED state)
  - weight (normally 1.0)
- Why are there weights?
  - Permits smooth blending of postures
- The PostureEngine class:
  - used to construct or store a posture
  - can take a “snapshot” of the robot's current state
  - can load from / save to a posture (.pos) file

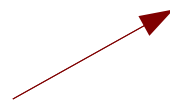
# Posture File: Simple Form

#POS

```
RFr:rotor      0.0000
RFr:sweep     -0.7402
RFr:elvtr     0.9454
RFr:knee      2.1954
LFr:sweep     0.3656
LFr:elvtr     0.8434
LFr:knee      1.9559
RMd:sweep     0.2077
RMd:elvtr     0.9346
RMd:knee      1.9206
LMd:sweep     0.1554
LMd:elvtr     0.8482
LMd:knee      1.8982
RBk:sweep     0.4219
RBk:elvtr     0.9658
RBk:knee      1.9334
LBk:sweep    -0.1051
LBk:elvtr     0.9052
LBk:knee      1.9633
```

```
ARM:elbow     0.0000
ARM:wristYaw  0.0000
ARM:wristPitch 0.0000
ARM:wristRoll 0.0000
ARM:gripper   0.0000
NECK:pan      0.0000
NECK:tilt     0.0000
LED:blue      0.0000
LED:green     0.0000
LED:yellow    0.0000
LED:red       0.0000
LED:RFr:knee  0.0000
LED:LFr:knee  0.0000
LED:RMd:knee  0.0000
LED:LMd:knee  0.0000
LED:RBk:knee  0.0000
LED:LBk:knee  0.0000
LED:NECK:pan  0.0000
```

#END



angle in radians

# Posture File: Condensed Form

Used by RawCamViewer when saving a snapshot.

**#POS**

condensed Chiara

meta-info = 59402 1856

outputs = 0 -0.199139 0.956064 2.31235 0.536929

0.805126 1.86361 0.0729802 0.950054 1.96743 0.0681147

0.856896 1.91196 0.212713 0.961196 1.93583 -0.387357

0.871718 1.85883 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

buttons = 0 0 0

sensors = 0 0 0 0 0 0 0 0 0 0 0

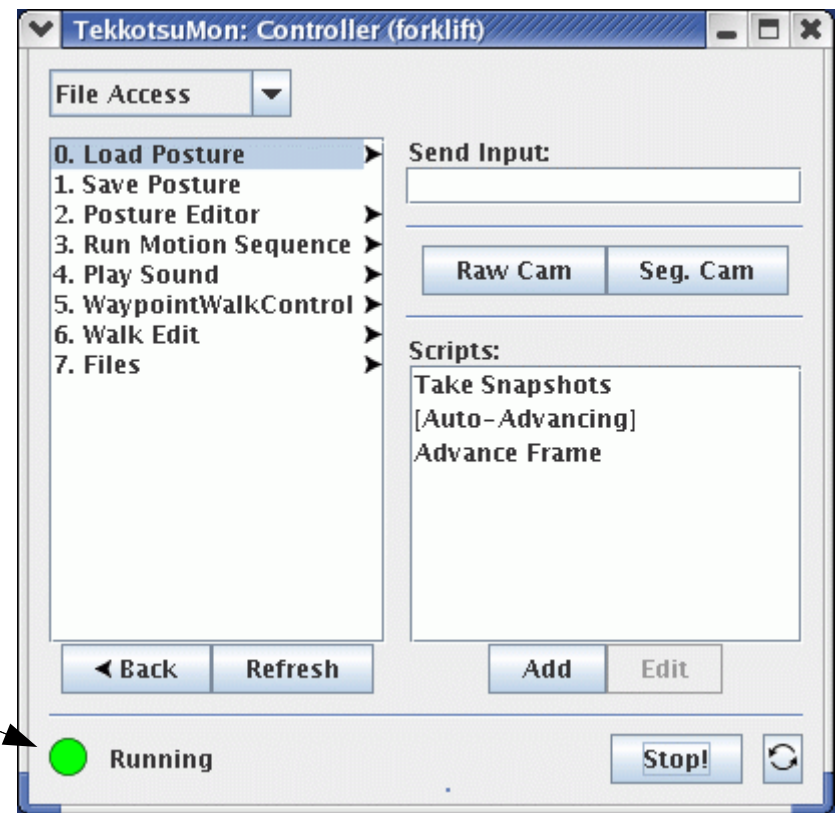
pidduties = 0

0 0 0 0 0

**#END**

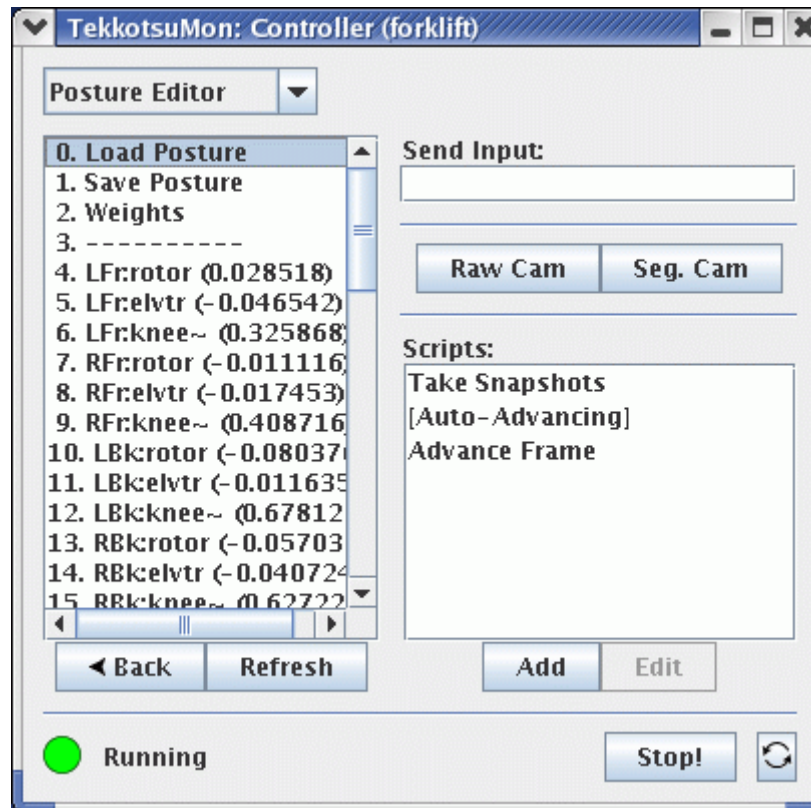
# Pre-Defined Posture Files

- Stored in project/ms/data/motion/\*.pos
  - For AIBO: stand.pos, situp.pos, liedown.pos, pounce.pos, rkick.pos
  - For Chiara: none yet
- Root Control > File Access > Load Posture
- Make sure Emergency Stop is off.



# Posture Editor

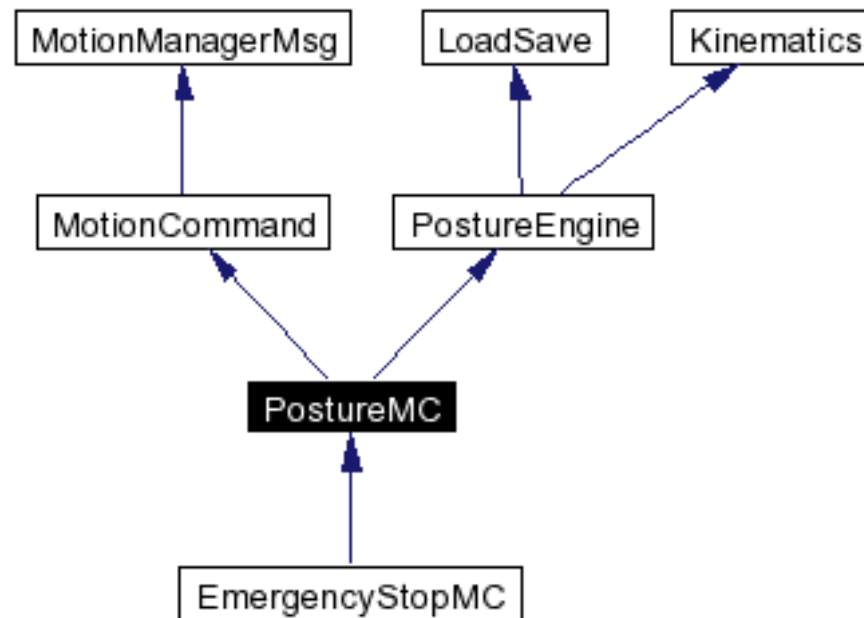
- Root Control > File Access > Posture Editor



- Turn on Relax mode and move the limbs.
- Save the file; then edit to remove irrelevant effectors.

# PostureMC

- PostureMC  $\equiv$  PostureEngine + MotionCommand



- Moves the effectors directly to the specified positions.
- Can optionally hold that position until deactivated.
- loadFile can be used to load a posture file.



# Chiara Effector Names

- Legs:

- LFrLegOffset, RFrLegOffset
- LMdLegOffset, RMdLegOffset
- LBkLegOffset, RBkLegOffset

*within each leg:*

- SweepOffset
- ElevatorOffset
- KneeOffset

e.g., LFrLegOffset+KneeOffset

Note: RFrLeg on the Chiara also has a rotator joint

## Head:

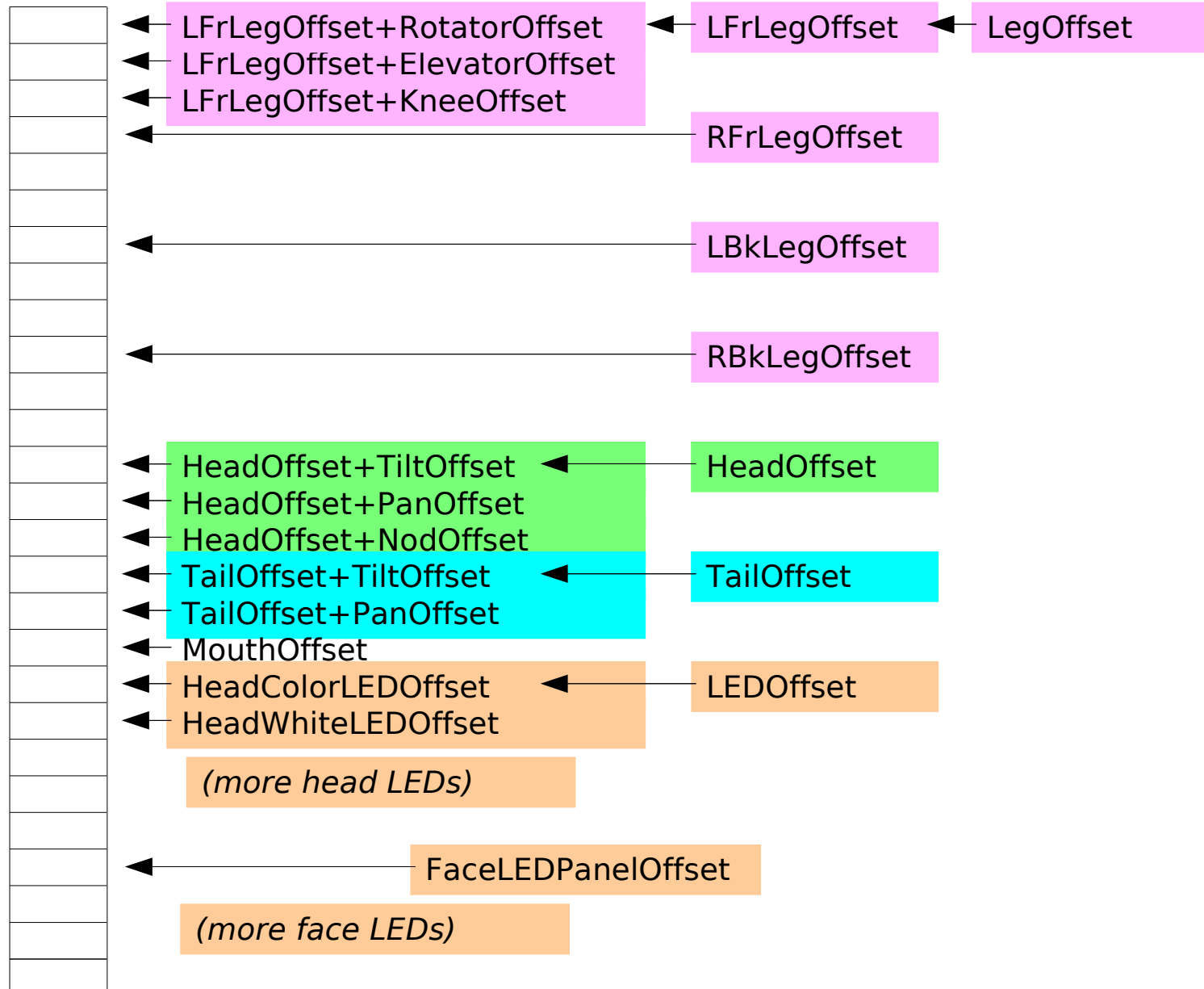
- HeadOffset
  - TiltOffset
  - PanOffset

e.g., HeadOffset+PanOffset

## Arm:

- ArmOffset
  - ArmShoulderOffset
  - ArmElbowOffset
  - WristOffset
    - WristYawOffset
    - WristPitchOffset
    - WristRollOffset
    - GripperOffset

# Effector Offsets (AIBO ERS-7)



# Sample PostureNode Code

- PostureNode contains a PostureMC.
- PostureMC inherits methods from both MotionCommand and PostureEngine; check documentation for both.
- Use a CompletionTrans =C=> to smoothly chain postures together.

```
startnode: PostureNode($, "lookleft.pos") =C=>  
           PostureNode($, "raiseLFrleg.pos")
```

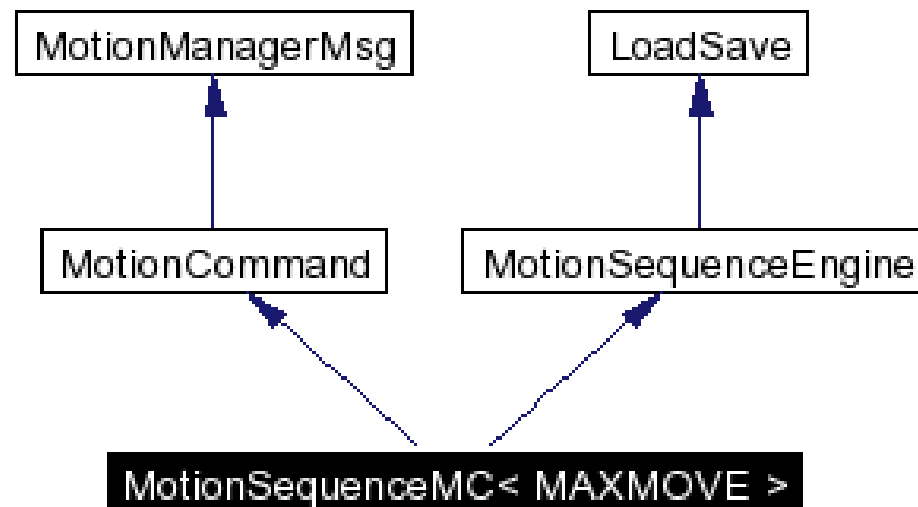
# Are We There Yet?

- PostureNode posts a completion event when the robot has been brought to the target posture.
- What if it never reaches the target?
  - Conflicting motion commands
  - Unreachable joint angles
  - Positioning error
- A timeout value tells the PostureMC when to give up.

# Motion Sequences

- Smoothly takes the robot through a sequence of postures, or “keyframes”.
- Each effector can be controlled independently.
- Since a MotionSequenceMC lives in shared memory, its size must be specified at compile time. (This is a relic from the AIBO's operating system.)
- `TinyMotionSequenceMC`  $\equiv$  `MotionSequenceMC<94>`
- Use `DynamicMotionSequence` to escape this limitation.

# MotionSequenceMC



# STANDLIE.MOT

- At time index 0, all joints are set to their current positions.
- Advance time index first, then specify target positions.
- MotionSequenceEngine will calculate joint velocities to achieve the specified targets at the appropriate times.

```
#MSq  
  
advanceTime 2000  
load stand.pos  
  
advanceTime 2000  
load situp.pos  
  
advanceTime 2000  
load liedown.pos  
  
#END
```

[See video](#)



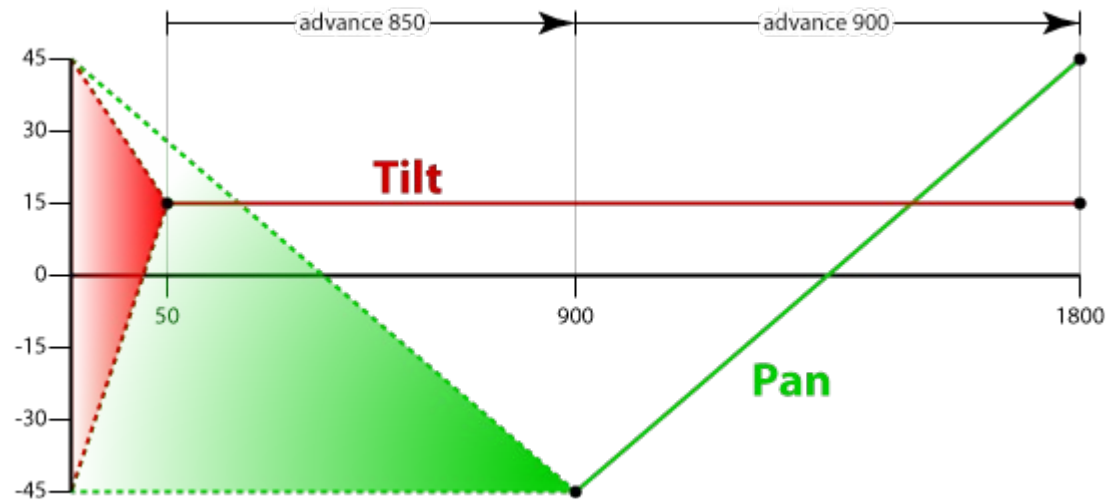
# PAN\_HEAD.MOT

#MSq  
degrees

advanceTime 50  
NECK:tilt 15  
NECK:nod~ 0

advanceTime 850  
NECK:pan~ -45

advanceTime 900  
NECK:pan~ 45  
NECK:tilt 15  
NECK:nod~ 0  
#END



Turn right 45°

Turn left 45°

Keep neck at 15°



See video



# HEADWAG.MOT

#MSq

degrees

advanceTime 50

NECK:pan~ 0

NECK:tilt 0

TAIL:pan~ 0

TAIL:tilt 0



Bring head and tail to  
neutral positions

advanceTime 1000

NECK:pan~ 90

Pan left

advanceTime 1000

NECK:pan~ -90

Pan right

advanceTime 500

NECK:pan~ 0

TAIL:pan~ 0

Center head

Update tail time index

advanceTime 500

TAIL:pan~ 90

Wag left

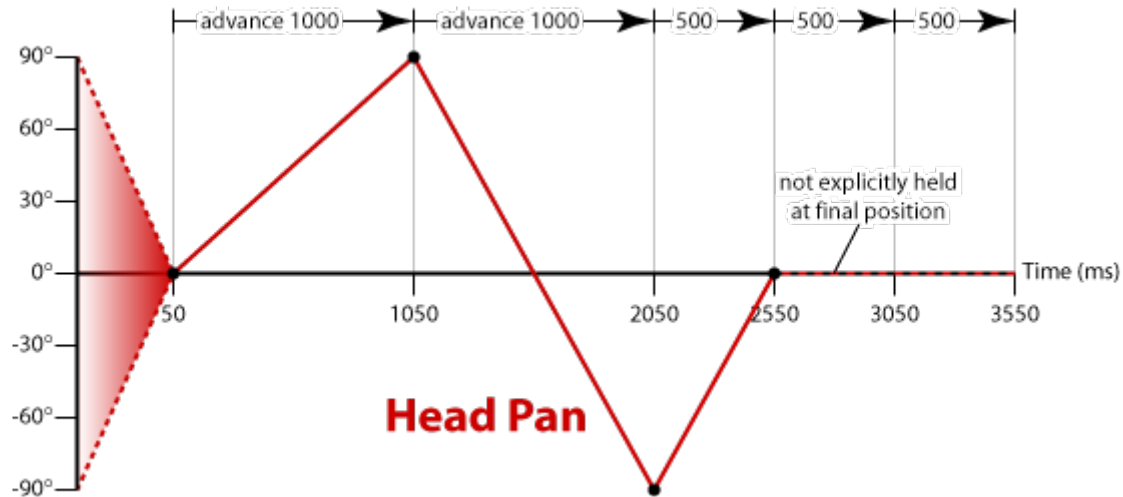
advanceTime 500

TAIL:pan~ -90

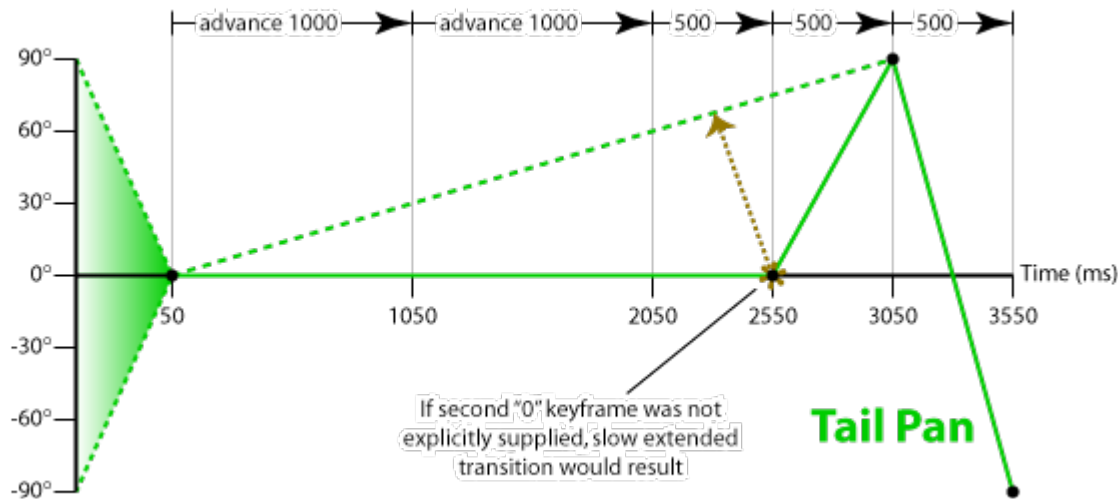
Wag right

#END

# HEADWAG.MOT



**Head Pan**



**Tail Pan**



**See video**

# Pre-Defined Motion Sequence Sizes

<u>Name</u>	<u># Full Postures</u>	<u># of Keyframes</u>
TinyMotionSequenceMC	2	94
SmallMotionSequenceMC	3	141
MediumMotionSequenceMC	6	282
LargeMotionSequenceMC	11	517
XLargeMotionSequenceMC	26	1222

# Jam Conditions

- Postures and motion sequences simply move the robot's effectors from current position to target position.
- They don't consider balance or friction.
- Problem #1: the robot can fall over.
- Problem #2: moving a leg when the robot's weight is on it can cause the motors to strain too hard, and “overload”.
- What's needed? Kinesthetic intelligence: the ability to reason about posture, balance, friction, and momentum.

# Lie down, Sit, Stand → Disaster

- Simple linear interpolation between stable postures is not guaranteed to produce stable transitions.



See video: [fallover.mp4](#)

- This is why kinematic intelligence is needed.