Postures and Motion Sequences

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

Carnegie Mellon
Spring 2012

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.

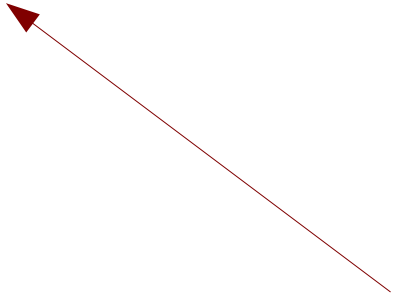
What is a “Posture”?

- A set of zero or more effector settings:
  - effector name (e.g., NECK:pan, or LED:Play)
  - 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
WHEEL:L              0.0000
WHEEL:R              0.0000
NECK:pan             0.0000
NECK:tilt            0.0000
ARM:base             0.0000
ARM:shoulder         0.0000
ARM:elbow            0.0000
ARM:wrist            0.0000
ARM:wristrot         0.0000
ARM:gripperLeft      0.0000
ARM:gripperRight     0.0000
LED:Power(Red)       0.0000
LED:Power(Green)     0.0000
LED:Play             0.0000
LED:Advance          0.0000
DesiredMode          1.0000
#END
```

Posture file for a Calliope5KP robot



angles in radians

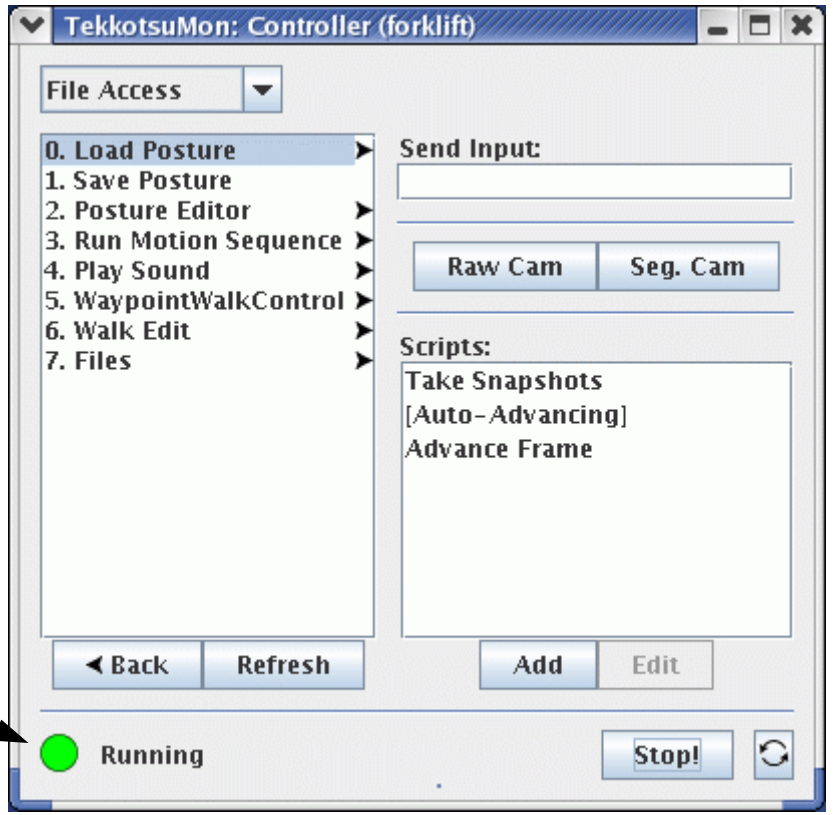
Posture File: Condensed Form

Used by RawCamViewer when saving a snapshot.

```
#POS
condensed Calliope5KP
meta-info = 667230 9584
outputs = 0 0 0 0 0 0 0 0 0 7.10543e-15 7.10543e-15 0 0 0 0 1
buttons = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
sensors = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
pidduties = 0 0 0 0 0 0 0 0 0 0 0 0
#END
```

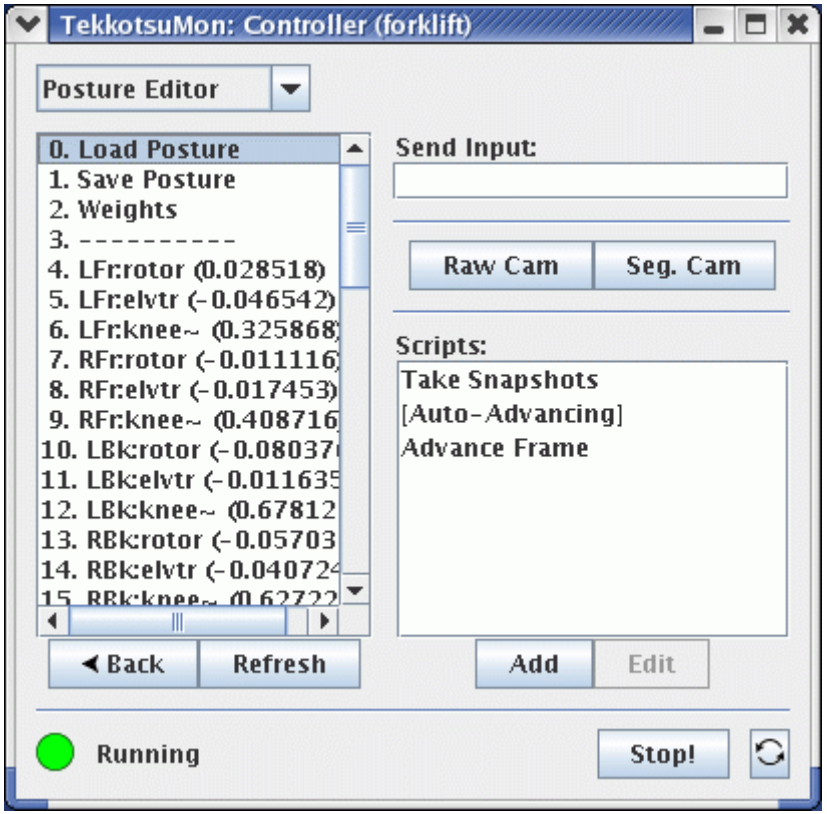
Pre-Defined Posture Files

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



Posture Editor

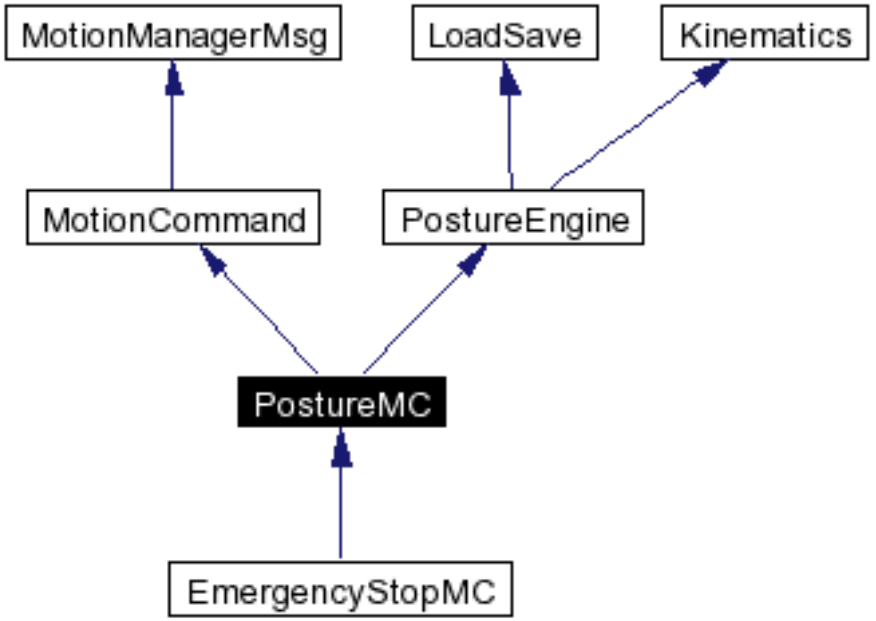
- Root Control > File Access > Posture Editor



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

PostureMC

- PostureMC ≡ 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.

Calliope Effector Names

Wheels: WheelOffset

Head:

- HeadOffset + ...
  - TiltOffset
  - PanOffset

LEDs: LEDOffset

- PowerRedLEDOffset
- PowerGreenLEDOffset
- PlayLEDOffset
- AdvanceLEDOffset

Arm: ArmOffset

- ArmBaseOffset
- ArmShoulderOffset
- ArmElbowOffset
- ArmWristOffset
- WristRotateOffset
- LeftFingerOffset
- RightFingerOffset

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:

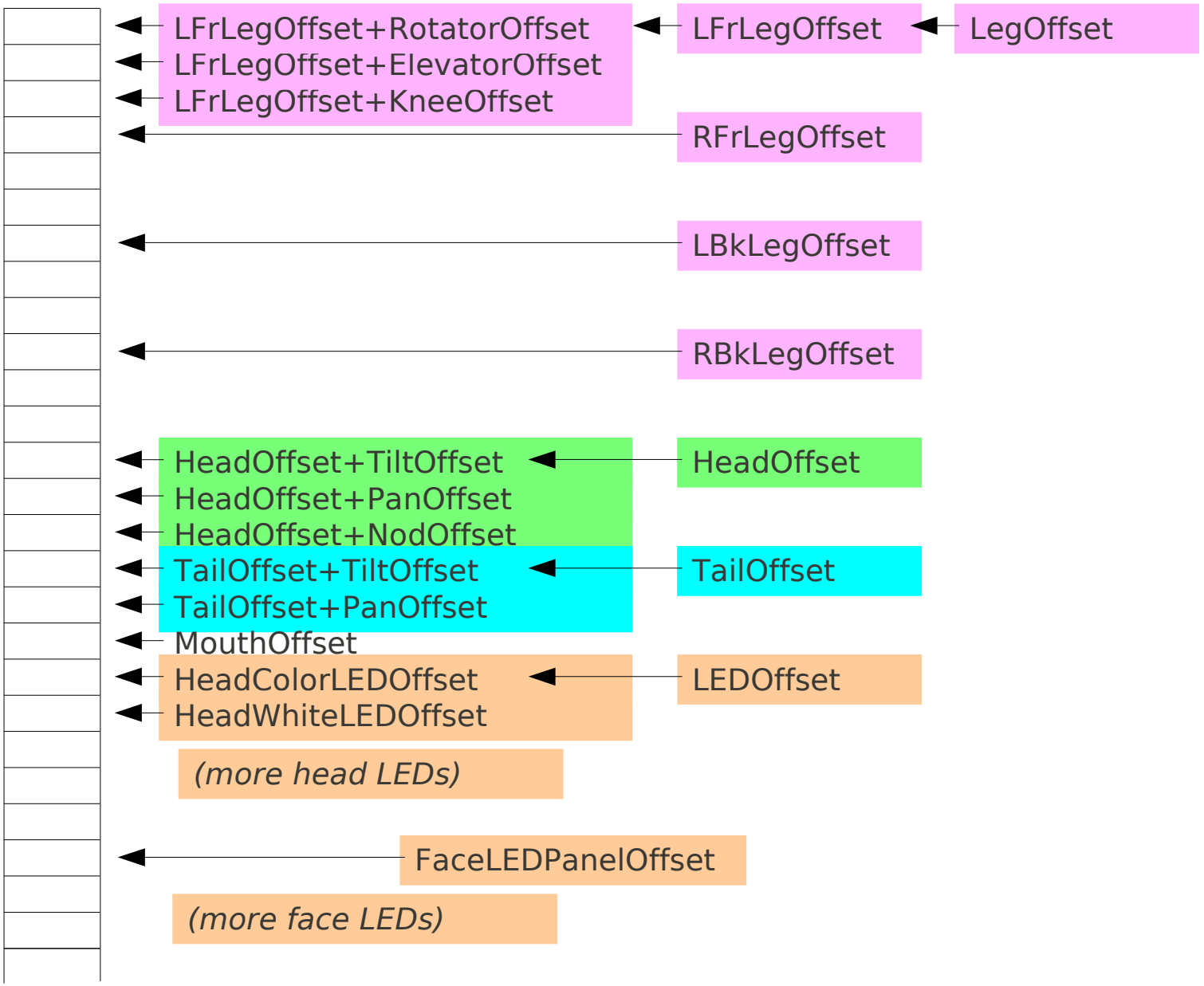
- HeadOffset
  - TiltOffset
  - PanOffset

e.g., HeadOffset+PanOffset

Arm:

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

Effector Offsets (AIBO ERS-7)



The RobotInfo File

- Tekkotsu/Shared/RobotInfo.h loads the info file for your model of robot.
- Tekkotsu/Shared/CommonCalliopeInfo.h
- Stores effector names and offset definitions.

```
const char* const sensorNames[NumSensors+1] = {  …

enum ArmOffset_t {
   ArmBaseOffset=ArmOffset,
   ArmShoulderOffset,
   ArmElbowOffset,
   ArmWristOffset,
   WristRotateOffset,
   LeftFingerOffset,
   RightFingerOffset
  };
```

Robot State Information

- The global variable `state` holds the current states of all the effectors.
- To access the elbow state:

```
state->outputs[ArmElbowOffset][
```

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”)
```

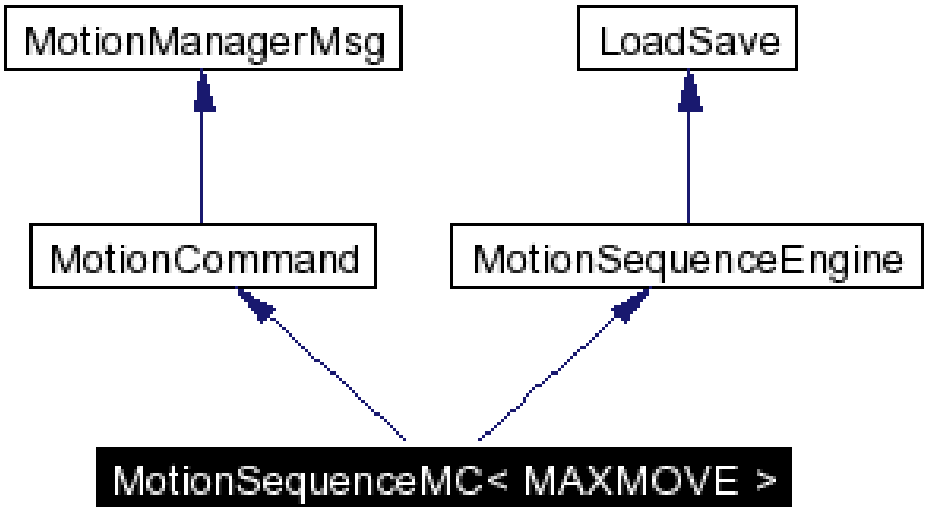
Are We There Yet?

- 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 ≡ 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

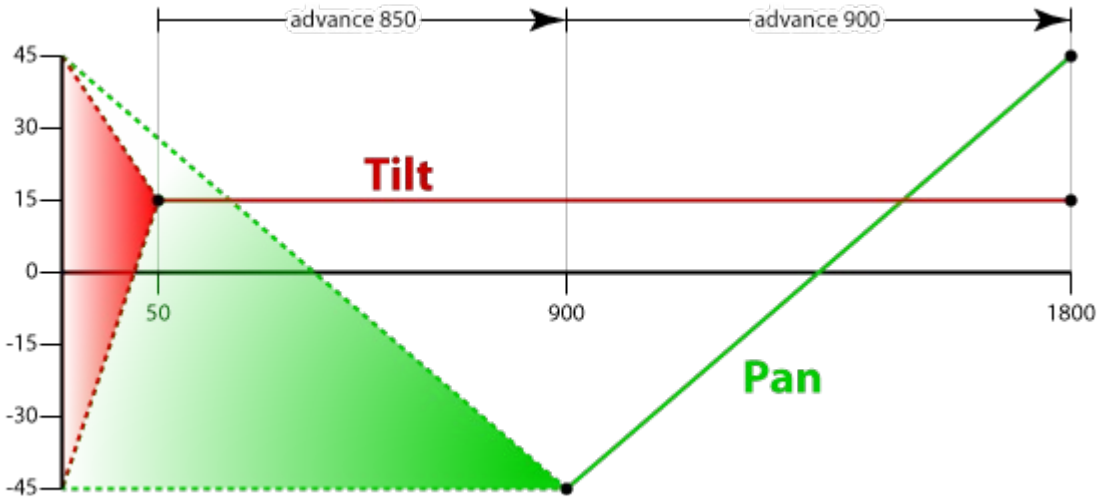
PAN_HEAD.MOT

```
#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

advanceTime 1000
NECK:pan~ 90

advanceTime 1000
NECK:pan~ -90

advanceTime 500
NECK:pan~ 0
TAIL:pan~ 0

advanceTime 500
TAIL:pan~ 90

advanceTime 500
TAIL:pan~ -90
#END
```

Bring head and tail to neutral positions

Pan left

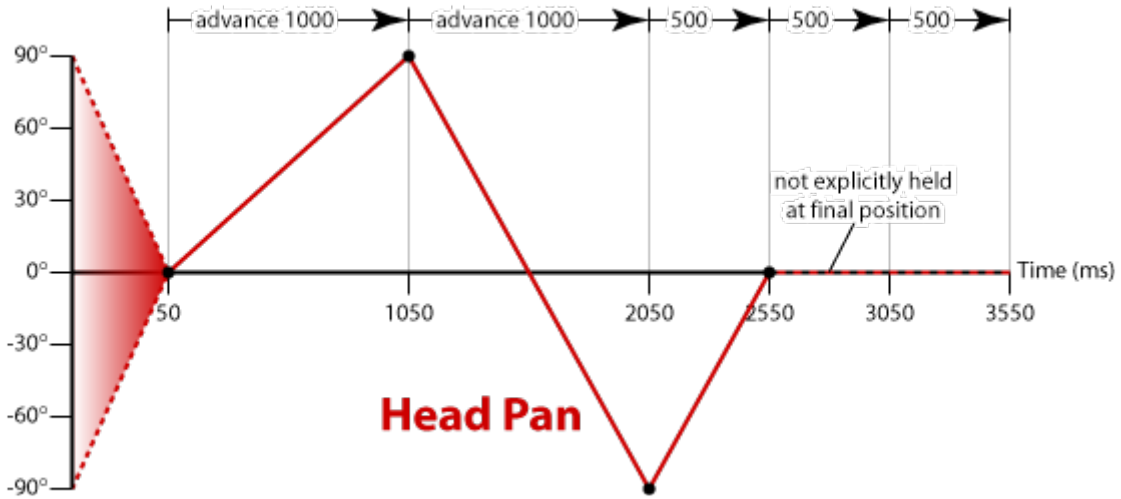
Pan right

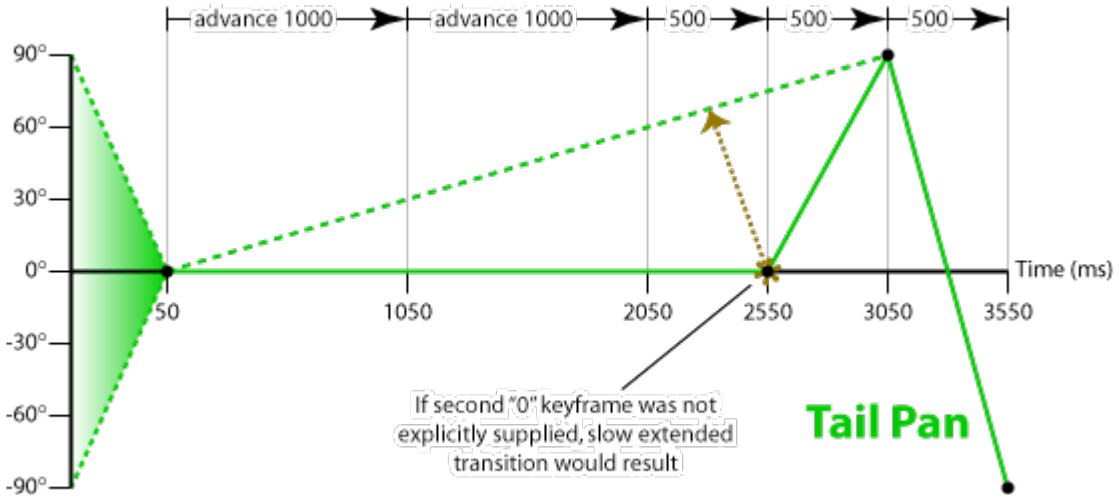
Center head
Update tail time index

Wag left

Wag right

HEADWAG.MOT





See video

Pre-Defined Motion Sequence Sizes

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

Don't bother with these. Use DynamicMotionSequence instead.

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.