Postures and Motion Sequences

15-494 Cognitive Robotics

David S. Touretzky &
Ethan Tira-Thompson

Carnegie Mellon

Spring 2008

How to Move the Body

1. Set joint angles directly with a Motion Command.
   - e.g., HeadPointerMC::setJointValue()
2. Specify a desired effect with a Motion Command.
   - e.g., WalkMC, HeadPointerMC::lookAtPoint()
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:rotor, or LED:faceK)
  - effector value (joint angle; LED state)
  - weight (normally 1.0)

- Why are there weights?

- 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 LIEDOWN.POS

```
#POS
LFr:rotor       0.946459        1.000000
LFr:elvtr       -0.034906       1.000000
LFr:knee~       0.602027        1.000000
RFr:rotor       0.924253        1.000000
RFr:elvtr       -0.052359       1.000000
RFr:knee~       0.585458        1.000000
LBk:rotor       -2.042035       1.000000
LBk:elvtr       0.245109        1.000000
LBk:knee~       1.978277        1.000000
RBk:rotor       -2.042035       1.000000
RBk:elvtr       0.233709        1.000000
RBk:knee~       2.100212        1.000000
#END
```

angle in radians

weight

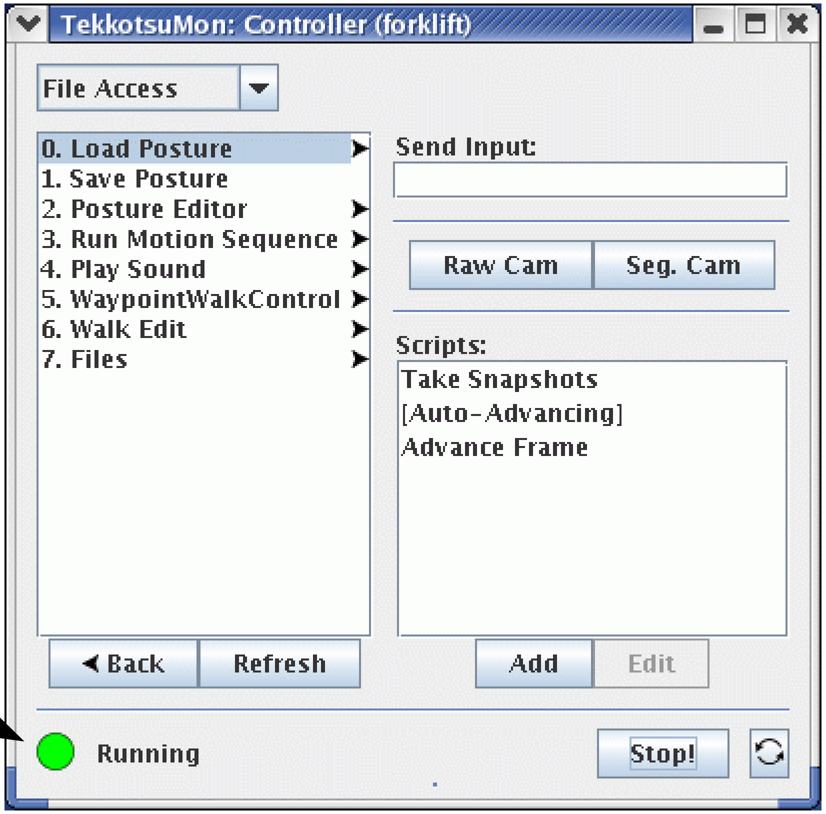
Posture File: Condensed Form

Used by RawCamViewer when saving a snapshot.

```
#POS
condensed ERS-7
meta-info = 410113 50176
outputs = -0.160775 0.193639 1.74154 -0.161567
           0.20503 1.75129 -0.800103 0.028476
           1.61991 -0.800441 0.022781 1.63301
          -0.037883 0.093161 -0.018371 0.017276
           1.0472 -0.064312 0 0 0 0 0 0 0 0 0 0
           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
buttons = 0 0 0 0 0 0 0 0 0 1
sensors = 450 700.288 126.836 1.18356 0.260353
          -9.12445 0.96 29.98 1944 7.89 -765
pidduties =  0 -0.0878906 -0.0683594 -0.0683594
            -0.107422 -0.0546875 0 0.0292969
            -0.0273438 0.0683594 0.078125 -0.123047
             0.0683594 0.0195312 0.0742188 0.0351562
            -0.00976562 0.0195312
#END
```

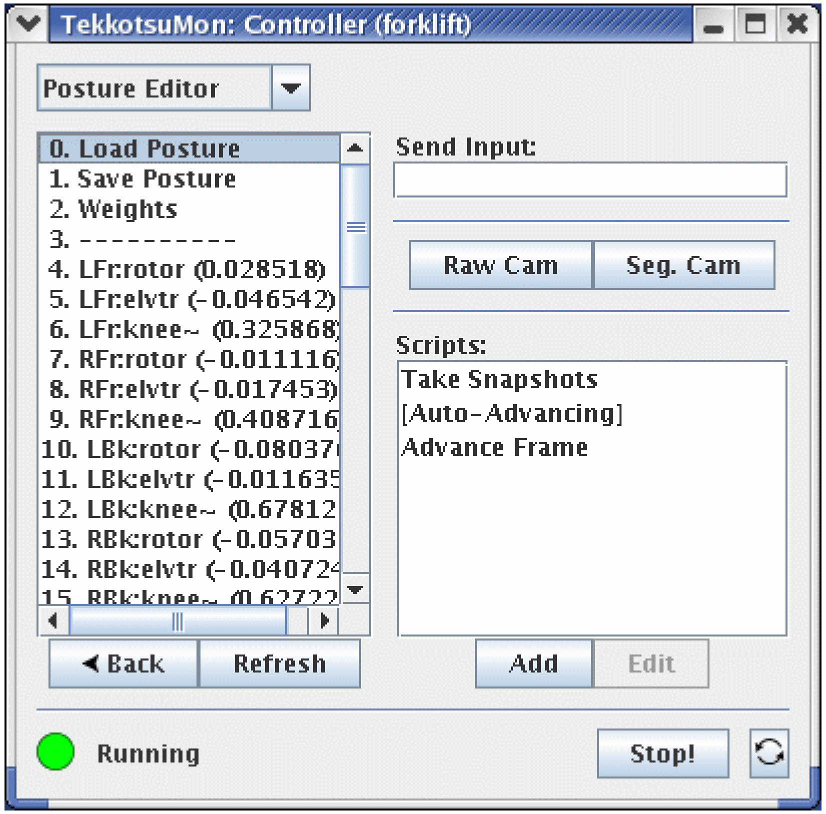
Pre-Defined Posture Files

- Stored in project/ms/data/motion/*.pos
  - stand.pos, situp.pos, liedown.pos
  - pounce.pos, rkick.pos
- Root Control > File Access > Load Posture
- Make sure Emergency Stop is off.



Posture Editor

- Root Control > File Access > Posture Editor



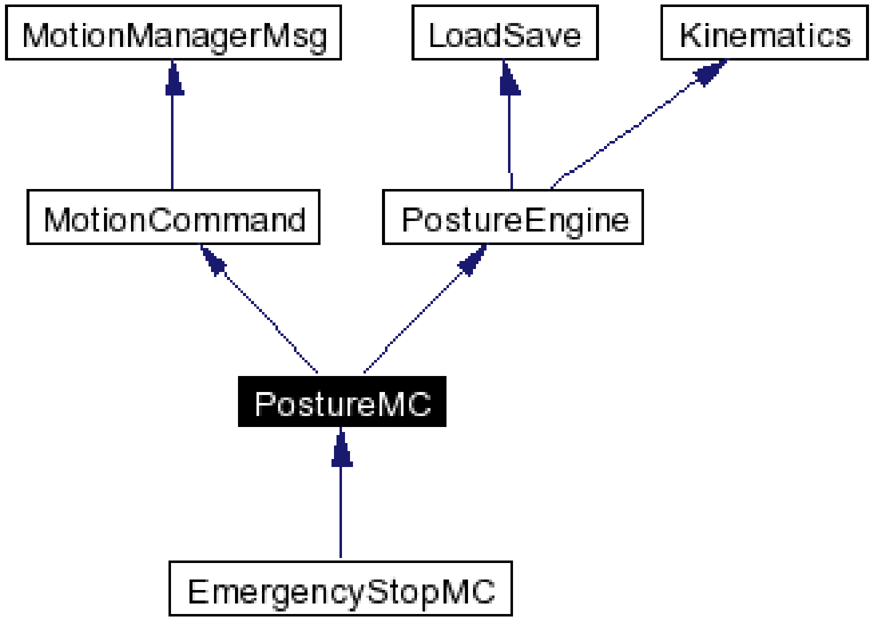
0. Load Posture
1. Save Posture
2. Weights
3. ----------
4. LFr:rotor (0.028518)
5. LFr:elvtr (-0.046542)
6. LFr:knee~ (0.325868)
7. RFr:rotor (-0.011116)
8. RFr:elvtr (-0.017453)
9. RFr:knee~ (0.408716)
10. LBk:rotor (-0.08037
11. LBk:elvtr (-0.011635
12. LBk:knee~ (0.67812
13. RBk:rotor (-0.05703
14. RBk:elvtr (-0.04072
15. RBk:knee~ (0.62722

Take Snapshots
[Auto-Advancing]
Advance Frame

- Turn *on* Emergency Stop and move the limbs.

PostureMC

- PostureMC ≡ PostureEngine + MotionCommand



- Moves the effectors directly to the specified positions.
- Can optionally hold that position until deactivated.
- One of its constructors takes a .pos filename argument.

Sample PostureMC Code

```
#include "Motion/PostureMC.h"

class MotionDemo : public BehaviorBase {
  private:
    SharedObject<PostureMC> pos_mc;
    MotionManager::MC_ID pos_id;

 public:
  MotionDemo() : BehaviorBase("MotionDemo"),
    pos_mc("stand.pos"), pos_id(MotionManager::invalid_MC_ID) {}

  virtual void DoStart() {
    BehaviorBase::DoStart();
    pos_id = motman->addPersistentMotion(pos_mc);
  }

  virtual void DoStop() {
    motman->removeMotion(pos_id);
    pos_id = MotionManager::invalid_MC_ID;
    BehaviorBase::DoStop();
  }
};
```

Effector Names

- Legs:
  - LFrLegOffset
  - RFrLegOffset
  - LBkLegOffset
  - RBkLegOffset

  within each leg:
  - RotatorOffset
  - ElevatorOffset
  - KneeOffset

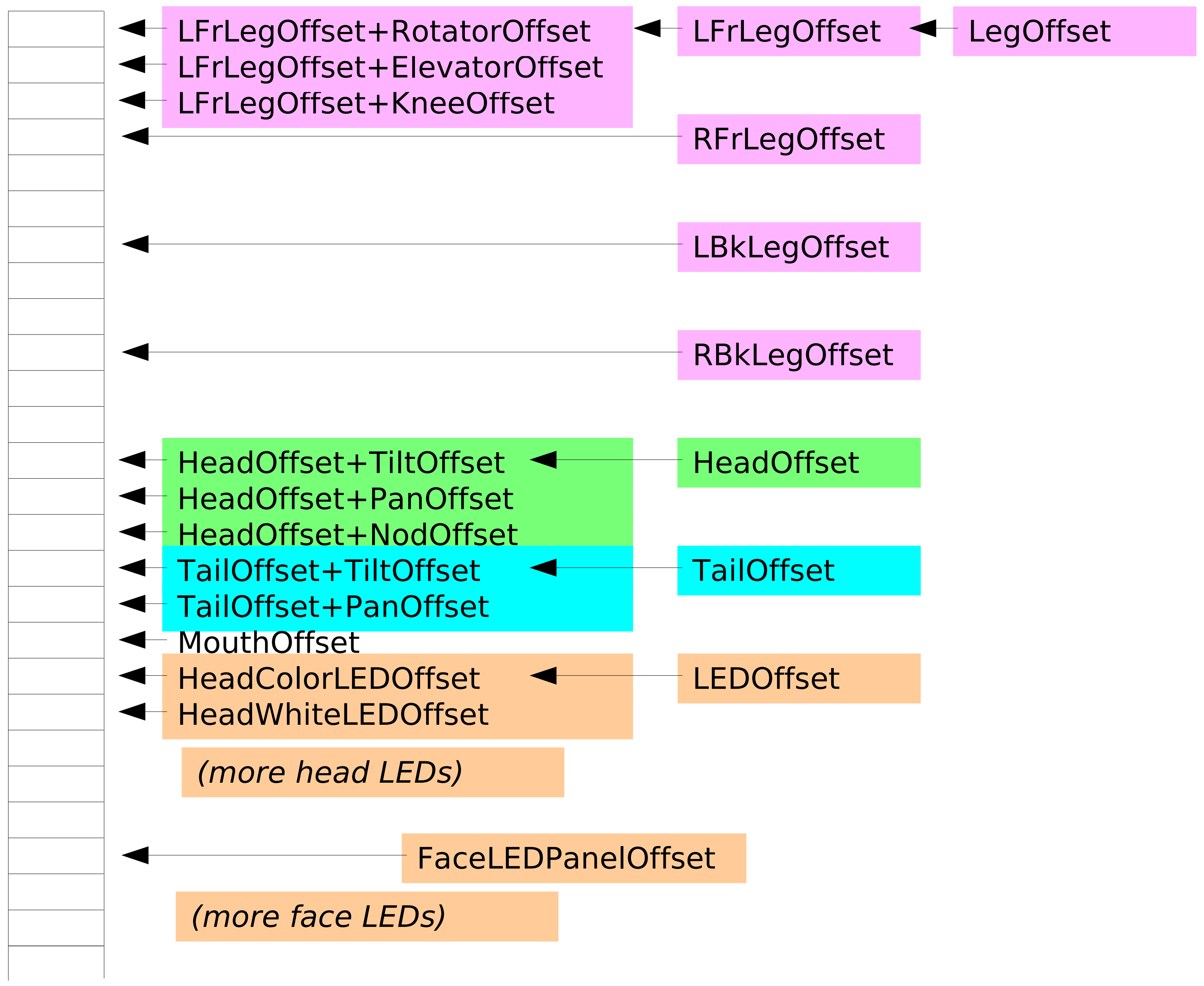
  e.g., LFrLegOffset+KneeOffset

- Head:
  - HeadOffset

  within the head:
  - TiltOffset
  - PanOffset
  - NodOffset

  e.g., HeadOffset+PanOffset
- Tail:
  - TailOffset
- *within the tail:*
  - TiltOffset, PanOffset

Effector Offsets



Setting Outputs Directly

setOutputCmd(unsigned int effector, float value)

The effector is specified by its index, e.g., HeadOffset+TiltOffset.

Sample code:

```
  virtual void DoStart() {
    BehavorBase::DoStart();
    pos_mc->setOutputCmd(HeadOffset+TiltOffset, 0.2);
    pos_id = motman->addPersistentMotion(pos_mc);
  }
```

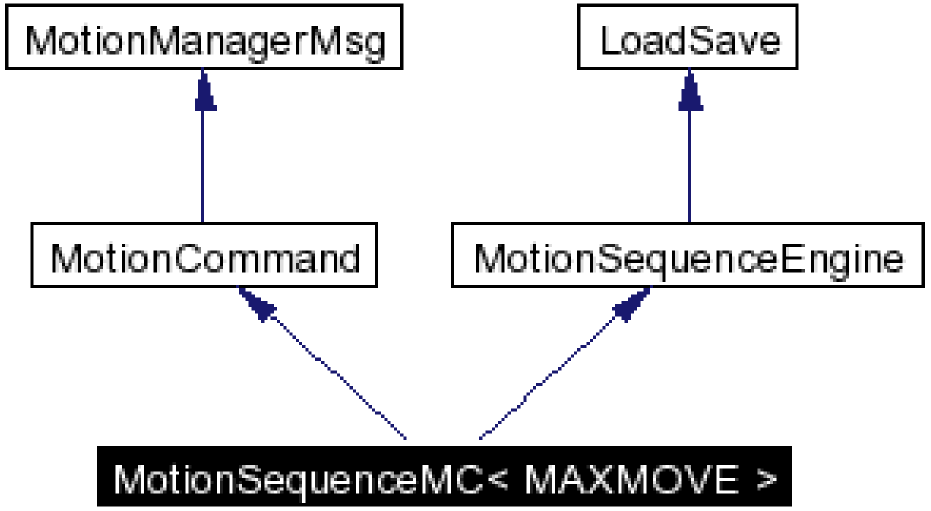
Are We There Yet?

- PostureMC posts a status 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.
- TinyMotionSequenceMC ≡ MotionSequenceMC<94>

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

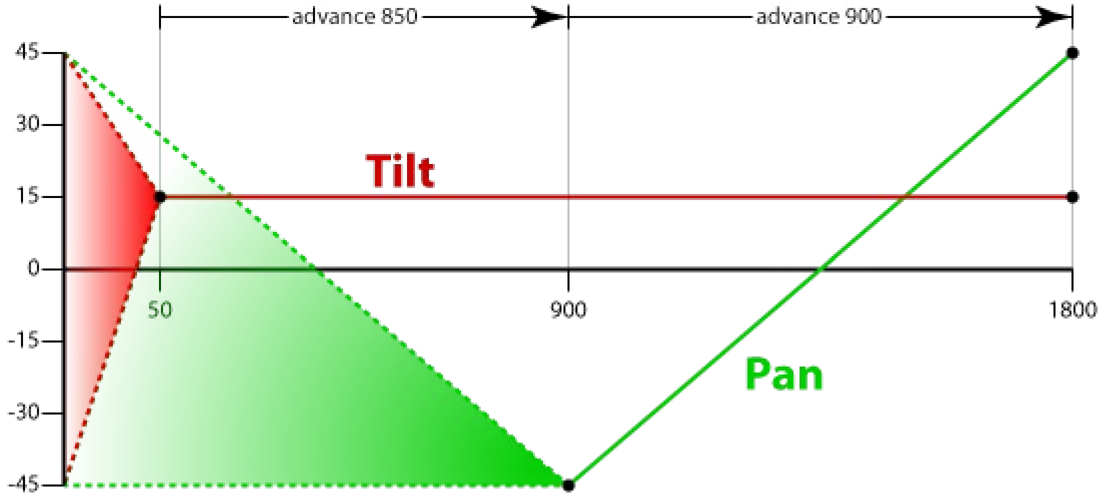
PAN_HEAD.MOT

```
#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

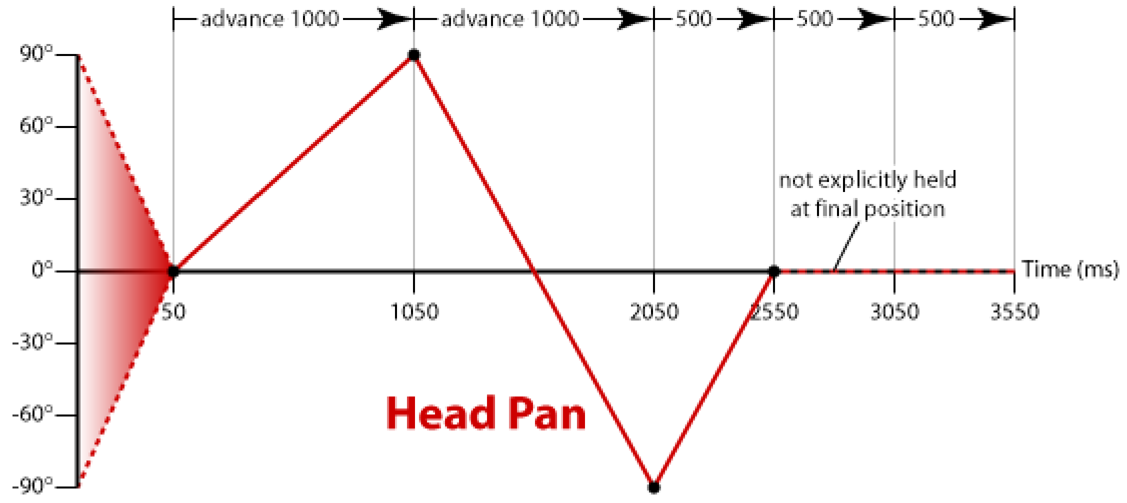
```
advanceTime 500
TAIL:pan~ 90
```

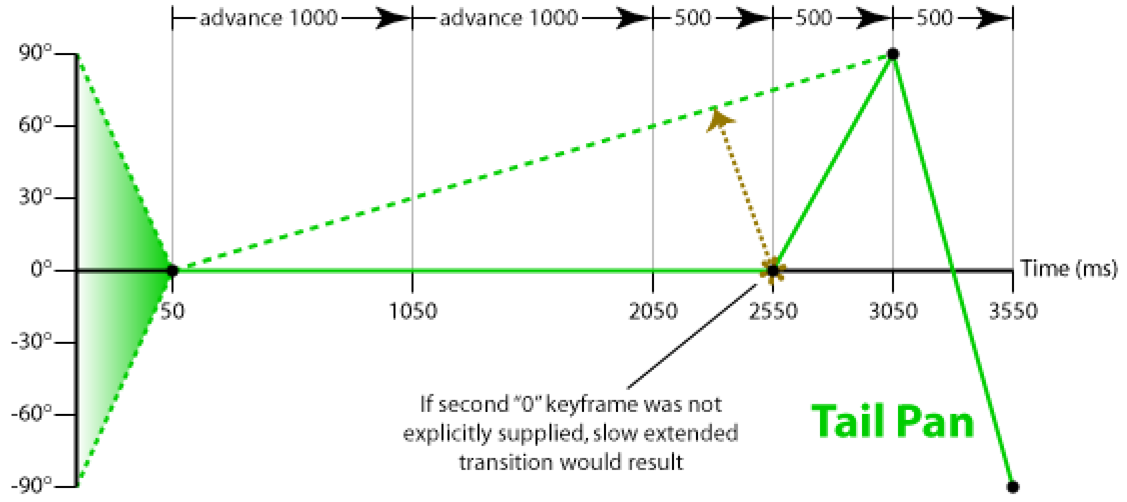
Wag left

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

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

Motion Sequence Example

```
#include "Shared/mathutils.h"     // for deg2rad

using namespace mathutils;

class DstBehavior : public BehaviorBase {
public:
  DstBehavior() : BehaviorBase("DstBehavior") {}

  virtual void DoStart() {
    BehaviorBase::DoStart();

    float const leftGlanceAngle = deg2rad(60.0);
    float const rightGlanceAngle = deg2rad(-70.0);
    float const downGlanceAngle = deg2rad(-55.0);
    SharedObject<MediumMotionSequenceMC> mseq_mc;
```

Motion Sequence Example (cont.)

```
// 1 sec to move head from current pos. to looking straight ahead
PostureEngine lookstraight;
lookstraight.setOutputCmd(HeadOffset+TiltOffset, 0.0);
lookstraight.setOutputCmd(HeadOffset+PanOffset, 0.0);
mseq_mc->advanceTime(1000);
mseq_mc->setPose(lookstraight);

mseq_mc->advanceTime(5000);    // 5 secs to sit up
mseq_mc->LoadFile("situp.pos");

mseq_mc->advanceTime(1000);    // 1 sec to glance left
mseq_mc->setOutputCmd(HeadOffset`+PanoffSet, leftGlanceAngle);

mseq_mc->advanceTime(2000);    // hold glance left for 2 secs
mseq_mc->setOutputCmd(HeadOffset+PanOffset, leftGlanceAngle);
```

Motion Sequence Example (cont.)

```
    mseq_mc->advanceTime(1000);    // 1 sec to glance right
    mseq_mc->setOutputCmd(HeadOffset+PanOffset, rightGlanceAngle);

    mseq_mc->advanceTime(2000);    // hold glance right for 2 secs
    mseq_mc->setOutputCmd(HeadOffset+PanOffset, rightGlanceAngle);
    mseq_mc->setOutputCmd(HeadOffset+TiltOffset, 0.0);

    mseq_mc->advanceTime(1000);    // 1 sec to glance down
    PostureEngine currentpose("situp.pos");  // update body joint time indices
    currentpose.setOutputCmd(HeadOffset+TiltOffset, downGlanceAngle);
    mseq_mc->setPose(currentpose);

    mseq_mc->advanceTime(5000);    // 5 secs to lie down
    mseq_mc->LoadFile("liedown.pos");
    mseq_mc->setOutputCmd(HeadOffset+TiltOffset, downGlanceAngle);

    motman->addPrunableMotion(mseq_mc);
    DoStop();
  }

};
```

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 flip over.
- Problem #2: moving a leg when the robot's weight is on it can cause the motors to strain too hard, and “jam”.
- What's needed? Kinesthetic intelligence: the ability to reason about posture, balance, friction, and momentum.

Lie down, Sit, Stand $\rightarrow$ Disaster

See video: fallover.mp4