



Sony's Aibo

Motion Commands & Postures

Robotics Seminar CSI445/660
Spring 2009, MW 4:15-5:35
Instructors: Tomek Strzalkowski,
Nick Webb & Michael Ferguson
University at Albany, SUNY



Administrative

- ASSESSMENT – pretend you don't remember ANYTHING I've said...
- HW1 due TODAY, email .h by midnight
- HW2 out now. Due next Wednesday
- Readings:
 - Browse tutorial parts on motion, postures



Getting Started

- **The Basic Problem [1]:**
 - We have n joints, each with a desired position which we have specified
 - Each joint has an actuator which is given a command in units of torque
 - Most common method for determining required torques is by feedback from joint sensors
- **PID (Proportional Integral Differential) Control**
- **The solution to this problem is based in forward and inverse kinematics for legged locomotion.**
- **Fortunately, Tekkotsu abstracts all of that for us**



Motion Commands

- Live in shared memory
- Are objects
- Methods for setting positions of joints
- An `updateOutputs()` called every 32ms by the motion manager



Lots of Motion Commands

- WalkMC
- HeadPointerMC
- TailWagMC
- LedMC



Creating a Motion Command

- `SharedObject<WalkMC> walk_mc;`
- The actual WalkMC object is created in shared memory.
- The SharedObject named walk_mc is created in main memory, holds pointer to actual WalkMC
- 2 ways to refer to it:
 - Via the SharedObject
 - Via the MC_ID assigned by the motman when the motion command is activated.

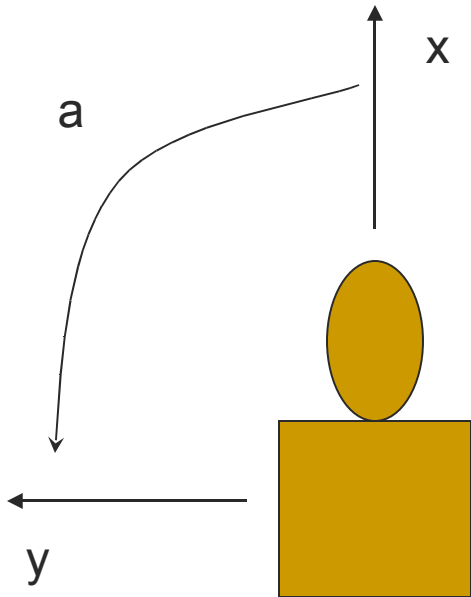


Using Motion Commands

- We need a unique ID:
`MotionManager::MC_ID walk_id;`
- Motion Commands are in shared memory, need to instantiate:
`walk_id = motman->addPersistantMotion(
 SharedObject<WalkMC>());`
- When we want to use the motion command, we need to “lock” it:
`MMAccessor<WalkMC> walk(walk_id);`



WalkMC



- WalkMC. Use the `setTargetVelocity` method to effect changes.
- `walkmc()->setTargetVelocity(x,y,a);`
 - x is positive in the forward direction
 - y is orthogonal to x. (left is positive)
 - a is the angular velocity and is the means by which to create turns.
 - positive in the counter-clockwise direction.
- `walkmc()->setTargetVelocity(150.0, 0, 0)` causes a straight forward walk
- `walkmc()->setTargetVelocity(0,0,0)` stops AIBO on the next update
 - up to 64ms lag time



Sample Behavior

```
#include "Behaviors/BehaviorBase.h"  
#include "Motion/WalkMC.h"  
#include "Motion/MotionManager.h"  
  
class WalkSample : public BehaviorBase{  
protected:  
    MotionManager::MC_ID walk_id;    // id of MotionCommand  
public:  
    WalkSample(): BehaviorBase("WalkSample"),  
                walk_id(MotionManager::invalid_MC_ID){}
```



Sample Behavior (cont'd)

```
Virtual void DoStart(){
    BehaviorBase::DoStart();
    std::cout << getName() << " is starting up." << std::endl;

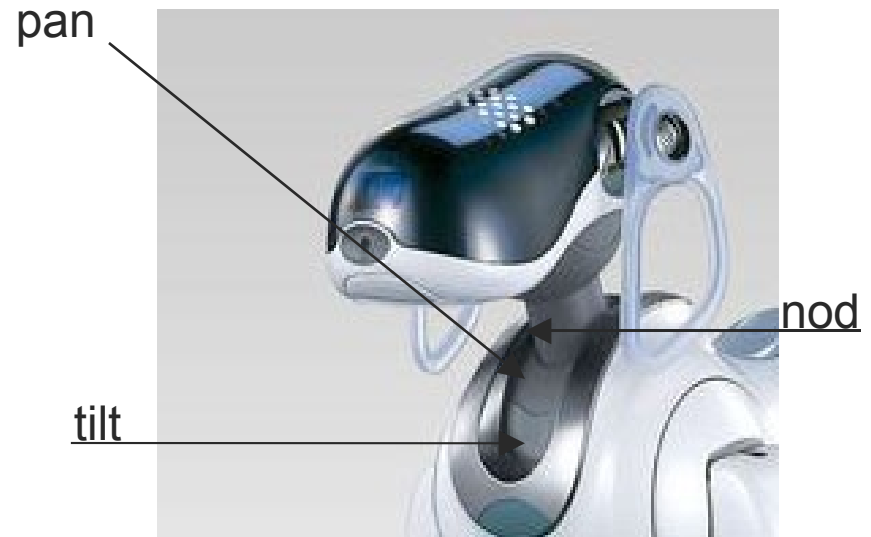
    walk_id = motman->addPersistantMotion(
        SharedObject<WalkMC>);
    MMAccessor<WalkMC>(walk_id)->setTargetVelocity(150,0,0);
}

virtual void DoStop() {
    motmon->removeMotion(walk_id);
    BehaviorBase::DoStop();
}
```



HeadPointer MC

- HeadPointerMC works very similarly to walkMC.
 - head.mc()->setJoints(tilt, pan, nod)
 - tilt – broad vertical joint
 - RAD(-75) to RAD(0)
 - pan – horizontal joint
 - RAD(-88) to RAD(88)
 - nod (roll on the 210) - fine vertical joint
 - RAD(-15) to RAD(45)





TailWagMC

- Uses a period (frequency) and magnitude to make the little tail wag
- Strictly a cuteness / realism MC
- Should this be implemented as a background behavior?
 - Depends on how long (or when) we want the tail to wag...



Postures

- A set of effector settings
 - Name, value, weight
- “Snapshot” of robot position.. load/save
- Stored in project/ms/data/motion/* .pos
 - stand.pos, situp.pos, pounce.pos
- Can be loaded in ControllerGUI
 - Root Control > File Access > Load Posture
-



Posture file

- Like a sequence, postures are created in a script file:
- The format is *extremely* simple:
 - Header line (mandatory)
 - #POS
 - One or more:
 - *Joint target-angle (rads) weight*
 - LFr:rotor -0.392146 1.000000
 - Close line (mandatory)
 - #END
- When the PostureEngine constructor is run it parses this file into a series of OutputCmd's which are executed (and averaged into other motion requests) when play() is called, or the Motion is added to motman (play() is implied there)



Using postures

```
#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_IC){}
```

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

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

```
};
```



Further Info

- http://cvs.tekkotsu.org/viewvc/Tekkotsu/Behaviors/Demos/DriveMeBehavior.cc?revision=1.11&view=markup&pathrev=tekkotsu-4_0