



# Motion

Robotics Seminar CSI445/660

Spring 2008

Robert Salkin

University at Albany, SUNY

# [ Review ]

- You can get data out of an event
  - `getGeneratorID`, `getSourceID`, `getTypeID`
  - Probably more reliable than checking `state->buttons[]` for a value
  - You want the event, not the value
    - The event means a button was pressed
      - May not still be pressed when the button value is checked
    - Not always true
      - Sensor values
      - Buttons acting as sensors
        - may be pressed harder or softer without activate/deactivate type events being triggered

# [ 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!!!

# MotionCommands

- Tekkotsu provides several primitive motions that can be used in our behaviors
  - WalkMC
    - Uses the CMPack engine to resolve and apply PID calculations on a 51 parameter set to achieve forward, reverse, and angled walking.
  - HeadPointerMC
    - Uses a three parameter set (tilt, pan, nod) to control the direction of Aibo's head.
  - TailWagMC
    - uses period, magnitude, and tilt to wag Aibo's tail.
- The first 2 are used in most behaviors

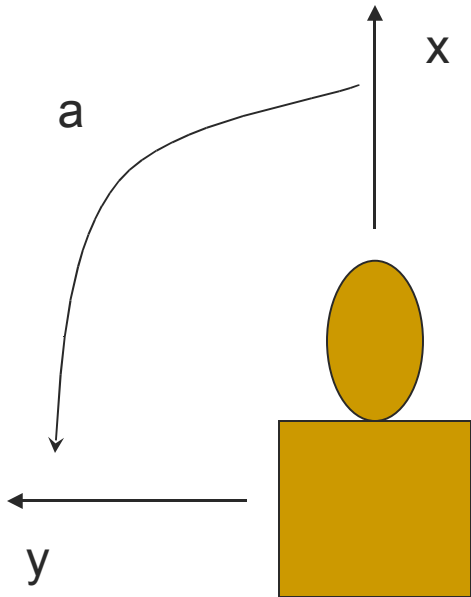
# [ Flags are helpful ]

- Use flags during event processing
  - `head=!head`
    - Flips value of flag when head is pressed
- Update physical state of AIBO only if an update is needed
  - Is the AIBO already stopped (or moving)?
- Changing the physical state in one place can avoid conflicting (or repetitive) instructions
  - Jerky movement

# Adding motion to a behavior

- Before diving into the structural details of writing our own MotionCommands, let's take look at some code to see how to implement these “black box” motions.
  - For each motion we wish to add, a unique ID needs to be created in our behavior and assigned for it
    - `MotionManager::MC_ID walk_id`
  - MotionCommands are part of the shared memory region, and need to be instantiated as such:
    - `walk_id = motman->addPersistentMotion(SharedObject<WalkMC>());`
  - Each motion needs a variable to provide access at the time a change needs to be made:
    - `MMAccessor<WalkMC> walk(walk_id);`

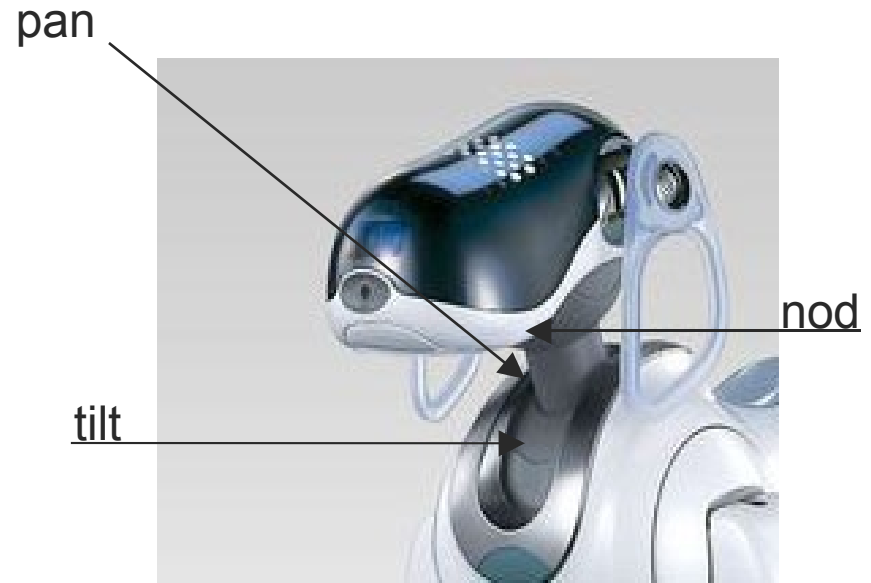
# [ WalkMC ]



- WalkMC. Use the `setTargetVelocity` method to effect changes.
- `walk.mc() -> 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.
- `walk.mc() -> setTargetVelocity(150.0, 0, 0)` causes a straight forward walk
- `walk.mc() -> setTargetVelocity(0, 0, 0)` stops AIBO on the next update
  - up to 64ms lag time

# 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...

# Deprecated addMotion

- `addMotion` is deprecated
  - `addPrunableMotion`
    - Motion that has a distinct endpoint
  - `addPersistentMotion`
    - Motion that has no distinct endpoint
- Listen for `motmanEGID` event with your `motion_id` as the `SourceID` to know when a motion ends and another may begin

# [ Important Links ]

- Walking Tutorial
  - [www.cs.cmu.edu/~dst/Tekkotsu/Tutorial/walking.shtml](http://www.cs.cmu.edu/~dst/Tekkotsu/Tutorial/walking.shtml)
- ERS7Info Namespace
  - <http://tekkotsu.org/dox/namespaceERS7Info.html>
- WorldState
  - <http://tekkotsu.org/dox/classWorldState.html>
  - For sensors, use:
    - `state->sensors [ERS7Info::field]`
- EventRouter::addTimer
  - <http://www.tekkotsu.org/dox/classEventRouter.html#5be35c4d722>
- See Tekkotsu.org for more documentation.

# [ References ]

---

- CMU Robobits Week 3 lecture “Motion”
  - <http://www-2.cs.cmu.edu/~robosoccer/cmrobobits/>
- Shawn Turner