



Sony's Aibo

Tekkotsu Basics FSM & Motion

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



Administrative

- HW1 is on site.. due Feb 4th
- Chiaras are HERE! (2 of them so far)
- HW2 will be out tomorrow, due Feb 11th
- Fan's office hours are on board
- I will not be here tomorrow afternoon



Readings

- **Tekkostu Tutorial**
 - Skim introduction, documentation sections
 - Behaviors
 - Events (skip Vision Events section)
 - Browse the states and transitions in Doxygen
 - Storyboard?

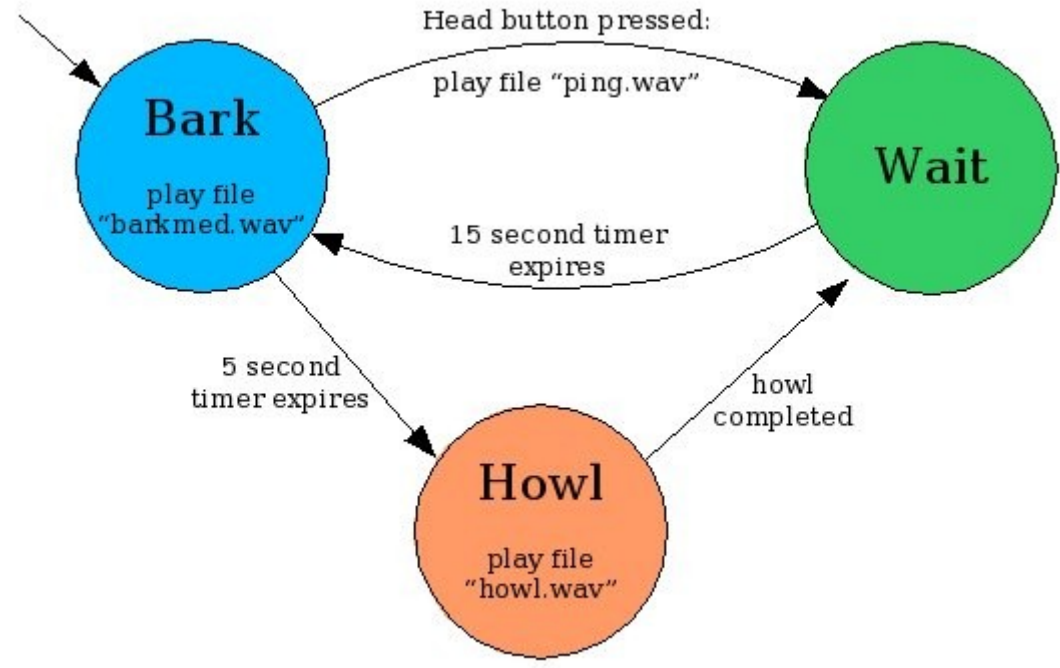


State Machines

- Simplest and most widely used robot control architecture.
- Easy to implement and understand.
- Not very powerful:
 - No dynamic planning
 - Everything is programmed explicitly.



Finite State Machines





Definition

- An abstract machine that defines a finite set of conditions of existence (called “states”), a set of behaviors or actions performed in each of those states, and a set of events that cause changes in states according to a finite and well-defined rule set.



Finite State Machines

- Efficient way to specify constraints of overall behavior of system
- Being in a state constrains which inputs the system responds to, and which states it can transition to
- FSMs allow decomposition of complex systems into modules (states) where transitions handle flow control.



Tekkotsu State Nodes

- In Tekkotsu, state machine nodes are behaviors.
- StateNode is child of BehaviorBase.
- To enter a state, call its start() method {which in turn calls DoStart()}
- To leave a state, call stop()
- StateNodes listen for and process events.



Key Functions

```
StateNode : BehaviorBase{  
    public:  
    DoStart()  
    DoStop()  
    addTransition(Transition *)  
    addNode(StateNode)  
    setup()  
    teardown()  
}
```



Lots of Nodes

- **StateNode**
 - SoundNode
 - WalkEngineNode
 - HeadPointerNode
 - WalkToTargetNode
 - TailWagNode
 - PostureNode



Transitions

- Are, you guessed it, subclass of BehaviorBase
- Transition's start() is called when its source state node becomes active
- Listens for events, fires..
- Deactivates its source and activates it's destination node(s).



Key Functions

```
Transition : BehaviorBase{  
    public:  
    DoStart()  
    DoStop()  
    fire()  
    processEvent(Event *)  
}
```



Lots of Transitions

- SmoothCompareTrans
- TimeOutTrans
- CompletionTrans
- EventTrans
- TextMsgTrans
- VisualTargetCloseTrans
- LostTargetTrans
- VisualTargetTrans



Nodes inside Nodes inside...

- We build our state machine (collection of nodes and transitions) inside a StateNode's *setup()* function
- *setup()* is called automatically the first time *start()* is called.
- *teardown()* destroys the state machine {@
DoStop()}



Example

```
#include "Behaviors/StateMachine.h"
```

```
class BarkHowlBlinkBehavior : public StateNode{  
public:  
    BarkHowlBlinkBehavior():  
        StateNode("BarkHowlBlinkBehavior"){}
```

```
...
```



StateMachine.h

- **A few words:**
 - Part of Tekkotsu 4.1
 - I have back-ported it to 4.0
 - Available on my SVN server:
 - svn.blunderingbotics.com



Registration

- Within the DoStart()
- Registering nodes:
 - `StateNode * node = new StateNode();`
`addNode(node);`
- Registering transitions
 - `node->addTransition(new TimeOutTrans(dest,time))`
- The variable *startnode* must point to the starting node.



And now...

- Walk through example...
- Discuss HW2