CMPE-013/L

Introduction to “C” Programming

Maxwell James Dunne
100

State machines
State machines

- Known as Finite State Machines (FSM)
- Mathematical model of computation where system has a single state
- Triggering conditions can change that state
- FSMs are defined completely by both their states and the transitions between them
State machines

State

• The system only exists in one state at a time
• State persists through time
• Certain conditions can change the state to another state
  – These are specific to the current state

Lock/bolt

Toaster oven

days

CMPE-013/L: “C” Programming
State machines

Translations

- Events trigger transitions between states
- A combination of events can be used
- Transitions are all mutually exclusive
- At any given time there must be a valid transition for a state
  - If no transition is explicitly stated, an implied loopback transition exists

Maxwell James Dunne
State machines

Benefits

• Provides a formal way to reason about a system
  – Allows for testing before writing any code
• Can be easily visualized
• Are language independent
• States are only dependent on current state and current inputs
State machines
When to use

• Can be used whenever there are a finite set of states for the system
  – Car transmission
  – Stoplight
  – Vending machine
  – Toaster oven
  – Video games
State machines
Use in the SeaSlug

• Transmission protocol
  – Mission management
  – Parameter management

• Operating state
  – Handling errors/system faults

• Calibration
  – Rudder
  – Radio controller
State machines

Diagrams

1. **STATE_1**
   - **condition2**
   - **action2**

2. **STATE_2**
   - **condition1**
   - **action1**
typedef enum { STATE_1, STATE_2 } SystemState;
static SystemState state;
{
    switch (state) {
    case STATE_1:
        default:
            if (condition1) {
                Action1();
                state = STATE_2;
            }
            break;
    case STATE_2:
        if (condition2) {
            Action2();
            state = STATE_1;
        }
    }
}
State machines

Integrating

Example

typedef enum { STATE_1, STATE_2 } SystemState;
static SystemState state;
int main (void) {
    // Initialize system

    // Event loop
    while (1) {
        // State machine
        switch (state) {
            ... 
        } 
    } 
}
CMPE-013/L

Bounce (or Hardware)

Maxwell James Dunne
Bounce

- Digital I/O
- A/D
- Timers
- Debouncing
AID 1023
0-100%
8 samples
Average
Buttons

0010100

up to down

down and Last event is not down

down event
debouncing \frac{1}{T}
4 buttons

1 button copy and paste
Button CEC
no blocking
no whiles
no fors

VN032