



State-Driven Agent Design

Artificial Intelligence for Interactive Media and Games

Professor Charles Rich
Computer Science Department
rich@wpi.edu

[Based on Buckland, Chapter 2 and lecture by Robin Burke]

IMGD 400X (B 08)

1

Outline (2 days)

- State machines
 - motivation
 - West World state examples
 - implementation code
- Messages
 - motivation
 - West World message examples
 - implementation code
- Advanced concepts
 - hierarchical state machines
 - non-deterministic state machines (Markov)
- Homework #2 – Bar Fly (due Sunday midnight)
- Review Chapter 3 (steering)
- Read/prepare Chapter 4 for next week (Simple Soccer)

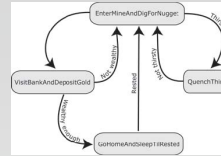


IMGD 400X (B 08)

2

(Finite) State Machines (FSM's)

- Familiar, easy to understand
 - standard graphical notation
 - good for communication
 - still most commonly used AI method in games
 - easy to combine with other methods (goals, etc.)
- Often very badly implemented
 - “spaghetti” code (if/then/else, switch, goto) --- a nightmare to maintain
 - we are going to study a generic object-oriented implementation



IMGD 400X (B 08)

3

West World

- A “laboratory” for studying FSM’s
 - no graphics -- simple plain-text to console
 - allows us to study all the code in detail
- Simulation-type game
 - two characters (agents): miner Bob and wife Elsa
 - next homework: add character Sal the bar fly
 - four locations: gold mine, bank, saloon, home
 - use FSM’s to model their activities

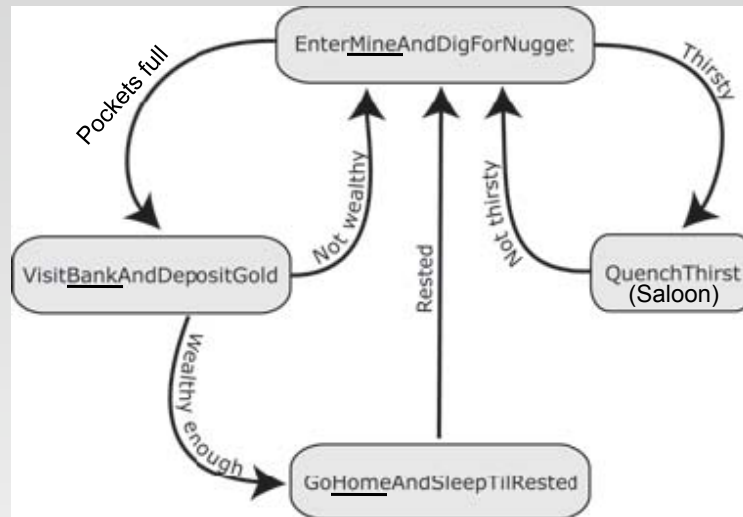
[get to do your own modeling in Homework #3]



IMGD 400X (B 08)

4

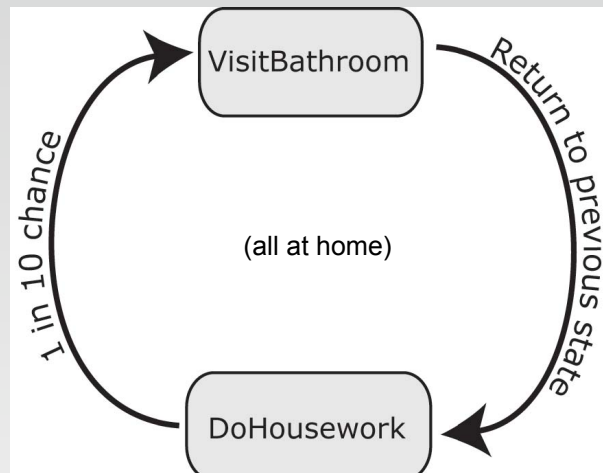
Miner State Machine



IMGD 400X (B 08)

5

Miner's Wife State Machine



IMGD 400X (B 08)

6

OO State Machine Implementation

- Each state is an **object**
 - encapsulates all information about the state
 - including how it decides which state (if any) to transition to next
 - generic template class, specific classes for game
 - *design issue*: states as singletons?
- Each agent has its own **state machine**
 - generic template class
 - current state
 - previous state (for “blips”)
 - global state (factor out shared code)



IMGD 400X (B 08)

7

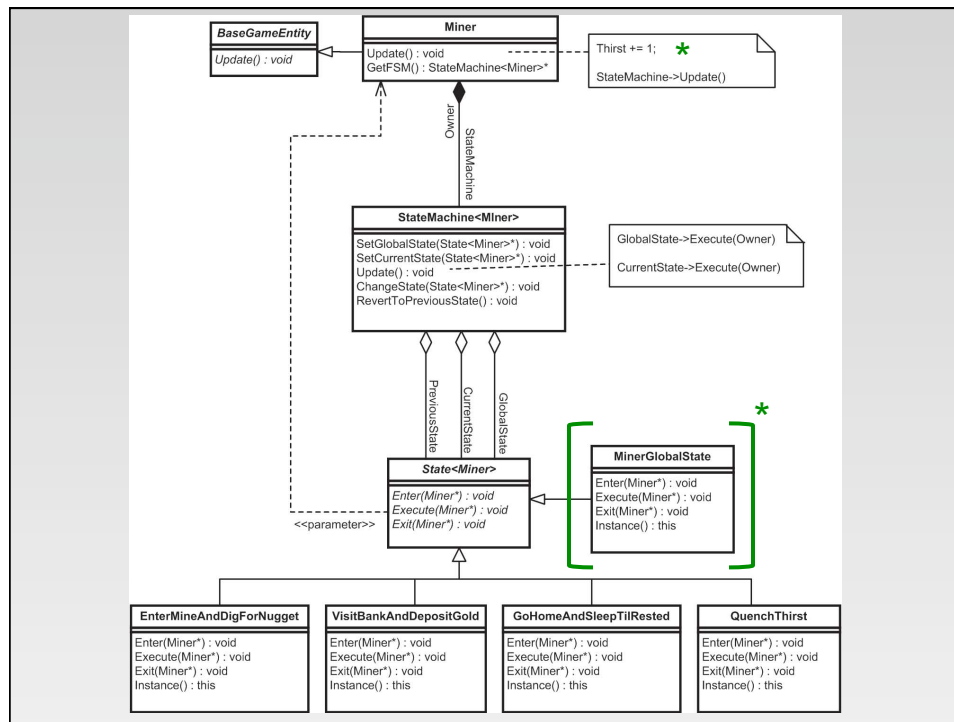
OO State Machine Implementation

- Calling sequence
 - game → agent: *“update yourself”*
 - agent → state machine: *“update yourself”*
 - state machine → current state:
 - “you are being entered for first time”*
 - “execute yourself”*
 - “you are being exited”*



IMGD 400X (B 08)

8



States as Singletons

- Each state class, e.g., QuenchThirst, has only a single instance
 - **Benefit:** don't need to manage allocation and destruction of state objects
 - **Drawback:** since all agents share same state objects, agent-specific information must be stored in agent (even if logically associated with state)
 - not a problem in West World, since only one miner, wife with distinct states
 - in Simple Soccer, states are also singletons, even though there are multiple players (agents) with same state machine

Singleton Design Pattern

```
// ----- MyClass.h -----
class MyClass
{
private:
    MyClass(){}
    ~MyClass(){}
    MyClass(const MyClass&);
    MyClass& operator= (const MyClass&);

    int m_iNum; // member data

public:
    static MyClass* Instance();
    int GetVal() const { return m_iNum; } // access data
}

// ----- MyClass.cpp -----
MyClass* MyClass::Instance()
{
    static MyClass instance;
    return &instance;
}

MyClass::Instance()->GetVal();
```



IMGD 400X (B 08)

11

Messaging – Why?

- Miner and wife in WWwW don't really interact
 - separate state machines running independently
 - states could “communicate” by shared variables
 - poor modularity
 - hard to add new agents which interact with existing
- A solution to the “perception” problem
 - avoids expensive polling algorithms (busy-wait)
 - e.g., if guard does nothing until player enters room, it should not be constantly be checking “did player enter” on every update cycle
 - instead, have player *send a message* to every entity in the room when she enters the room
- Modern games use messaging extensively



IMGD 400X (B 08)

12

Messaging - Implementation Issues

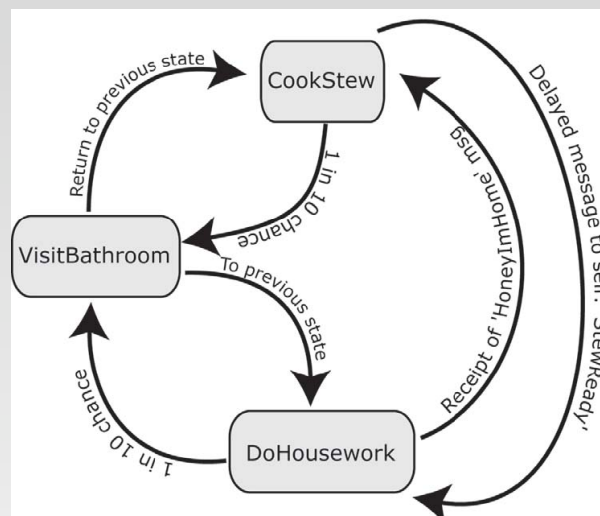
- Requires unique id registry for every participating entity
 - see BaseGameEntity and EntityRegistry
- Different delivery variations
 - *point-to-point* (messages addressed to specific recipients) -- as in Buckland code
 - *broadcast* (all messages broadcast to all entities --- expensive)
 - *subscription based on*
 - location (e.g., room)
 - message type
 - *delayed delivery* -- Buckland



IMGD 400X (B 08)

13

Miner's Wife State Machine (extended)



IMGD 400X (B 08)

14

West World Message Types

■ **HiHoneyImHome**

- sent by **Bob** to **Elsa** when entering **GoHomeAndSleepTilRested** state
- **Elsa** responds in **WifesGlobalState** by changing state to **CookStew**

■ **StewReady**

- sent by **Elsa** to self (with delay) when entering **CookStew** state
- **Elsa** responds in **CookStew** state by sending **StewReady** message (note reuse) to **Bob**
- **Bob** responds in **GoHomeAndSleepTilRested** state by changing state to **EatStew** (blip)



IMGD 400X (B 08)

15

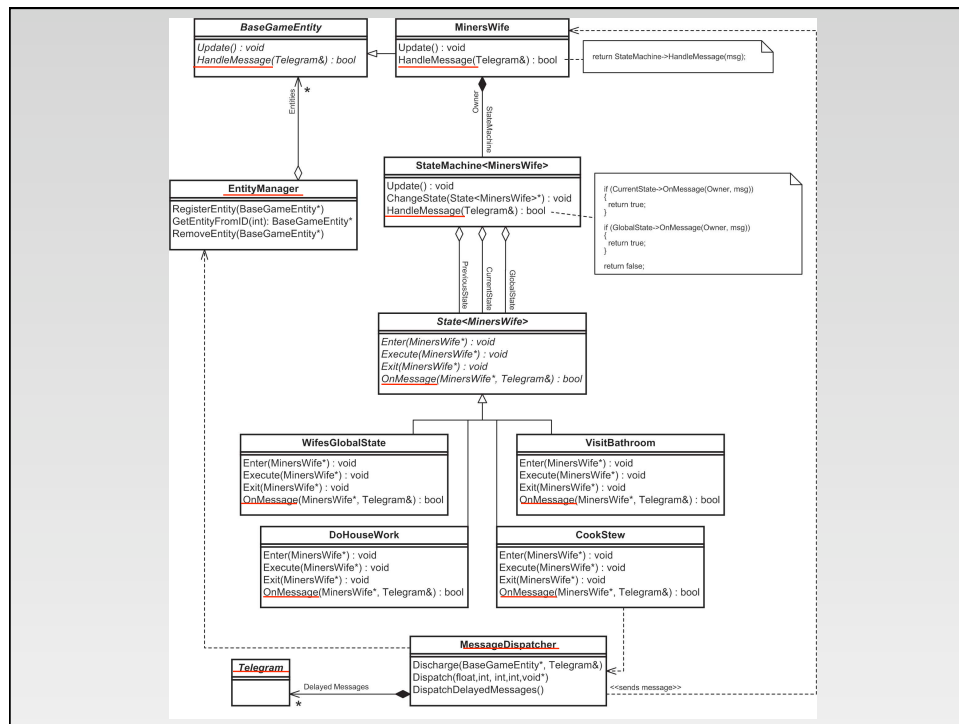
WestWorldWithMessaging Demo

- Various text strings printed to console by Elsa and Bob at various points, e.g.
 - “putting the stew in the oven”
 - “smells Reaaal goood Elsa!”
 - don’t confuse these “messages” with MessageType’s
- This is *programming!*
 - with all the bugs and debugging
 - if a message not handled properly or ignored, whole simulation can stall



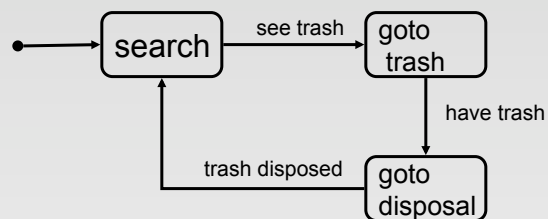
IMGD 400X (B 08)

16

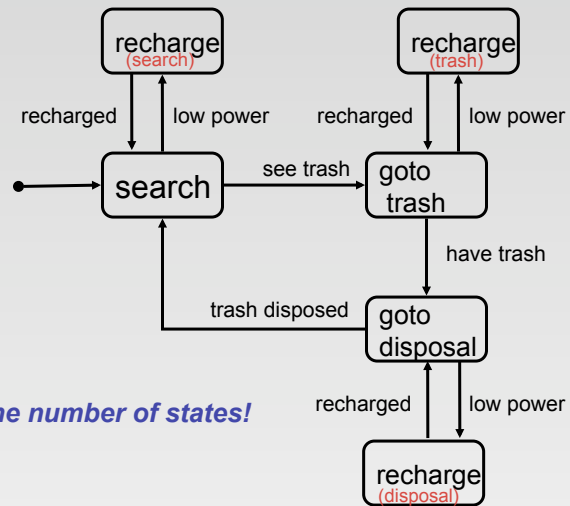


Hierarchical State Machines

Why?



Interruptions (e.g., Alarms)



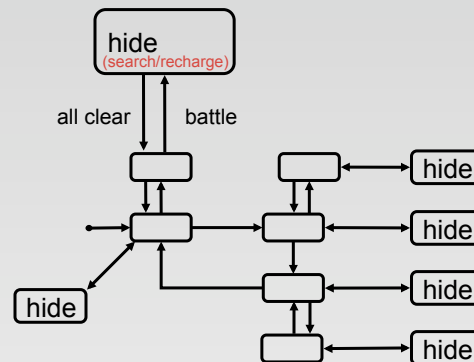
6 - doubled the number of states!



IMGD 400X (B 08)

19

Add Another Interruption Type



12 - doubled the number of states again!

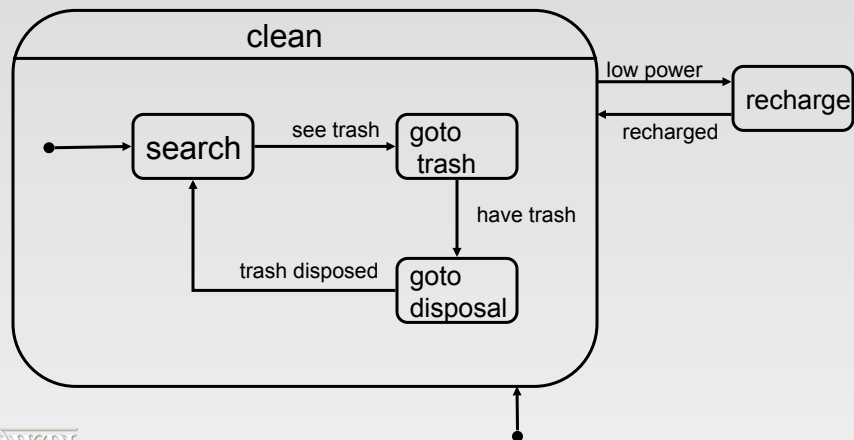


IMGD 400X (B 08)

20

Hierarchical State Machines

- leave any state in (composite) 'clean' state when 'low power'
- 'clean' remembers internal state and continues when returned to via 'recharged'

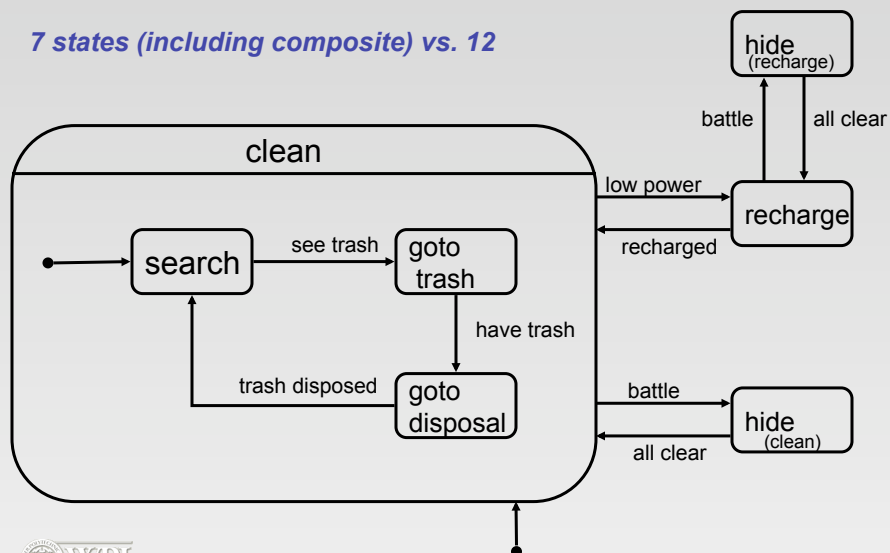


IMGD 400X (B 08)

21

Add Another Interruption Type

7 states (including composite) vs. 12



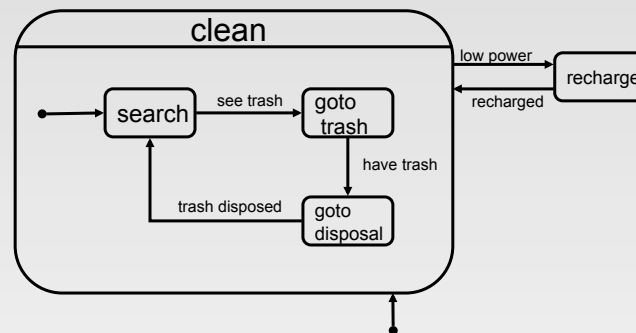
IMGD 400X (B 08)

22

Cross-Hierarchy Transitions

Why?

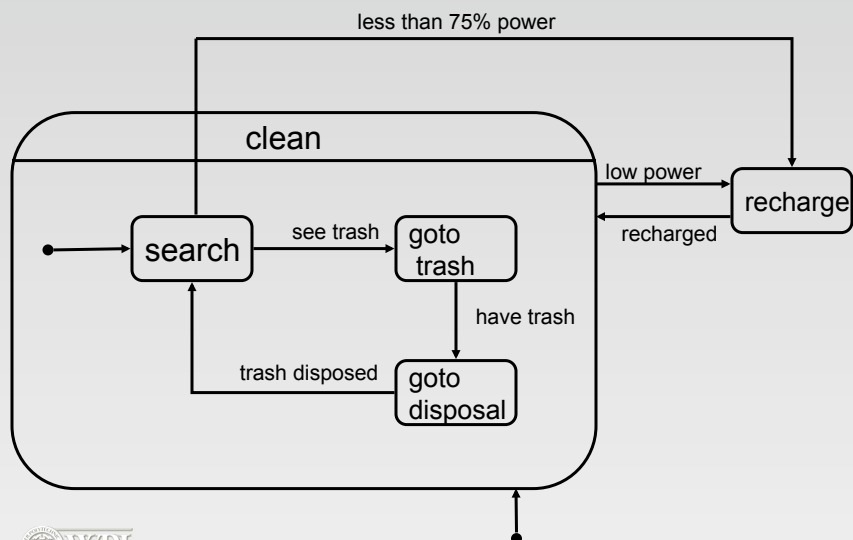
- suppose we want robot to top-off battery if it doesn't see any trash



IMGD 4000X (B 08)

23

Cross-Hierarchy Transitions

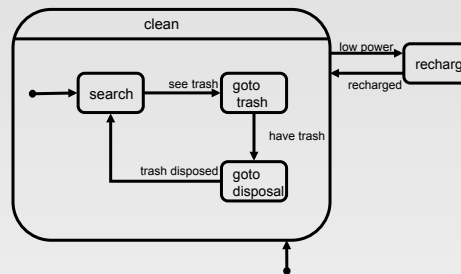


IMGD 400X (B 08)

24

Hierarchical State Machines

- 'Blip' states in Buckland implementation are simple case (remembers single previous state)
- General case has full push-down stack
- See Millington Sec. 5.3.9 for more details

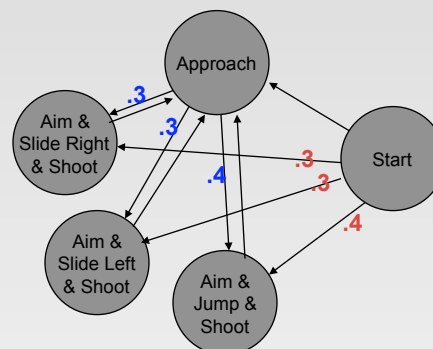


IMGD 400X (B 08)

25

Non-deterministic State Machines

- multiple transitions for same event
- label each with probability ($\Sigma=1$)
- state machine randomly chooses at run time, based on probabilities
- adds variety to actions

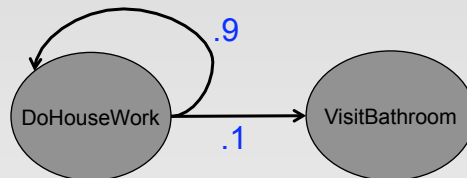


IMGD 400X (B 08)

26

Non-deterministic State Machines

- Also known as "Markov Models"
- Similar effect achieved in miner's wife states using ad hoc code rather than general machine



- See Millington, Sec. 5.5.2 for more details
- Similar variety effect can also be obtained with fuzzy logic (Chapter 10)



IMGD 400X (B 08)

27

Coming up...

- Homework #2 – Bar Fly (due Sunday midnight)
 - adding another character/agent to West World
 - new states and messages
- Review Chapter 3 (steering)
- Start reading Chapter 4 to prepare for next week (Simple Soccer)



IMGD 400X (B 08)

28