FSM - Introduzione

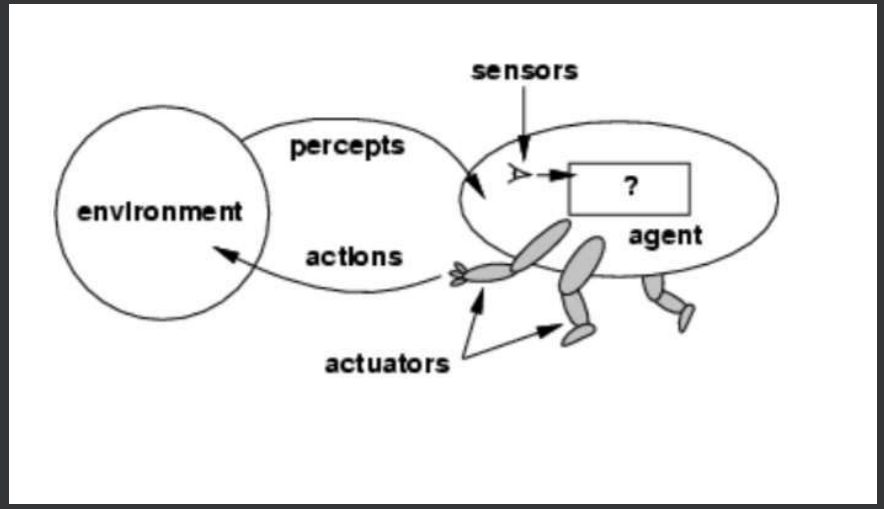
CORSO DI GAME PROGRAMMING
1° ANNO

Docente **Davide Caio**

Agenti ed Environment

Si definisce **AGENTE** un sistema integrato in un **AMBIENTE** (Environment) capace di:

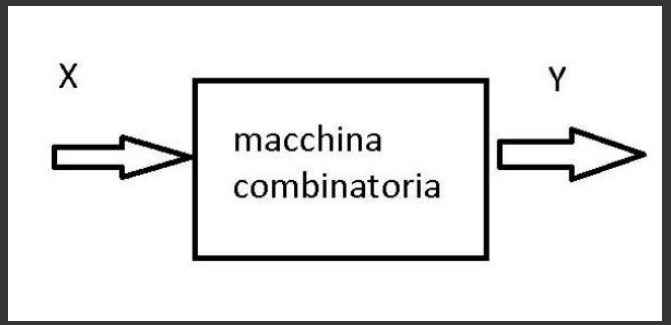
- Osservare lo stato dell' environment
- Avere una propria percezione di tale stato
- Agire in maniera proattiva in conseguenza delle proprie percezioni



Logiche Combinatorie

Si definisce logica combinatoria un qualsiasi sistema che, dato un insieme finito di possibili input, in un istante di tempo T restituisce un output dipendente solo dall'ingresso in tale istante T.

Un sistema di questo tipo di dice anche **MACCHINA COMBINATORIA**

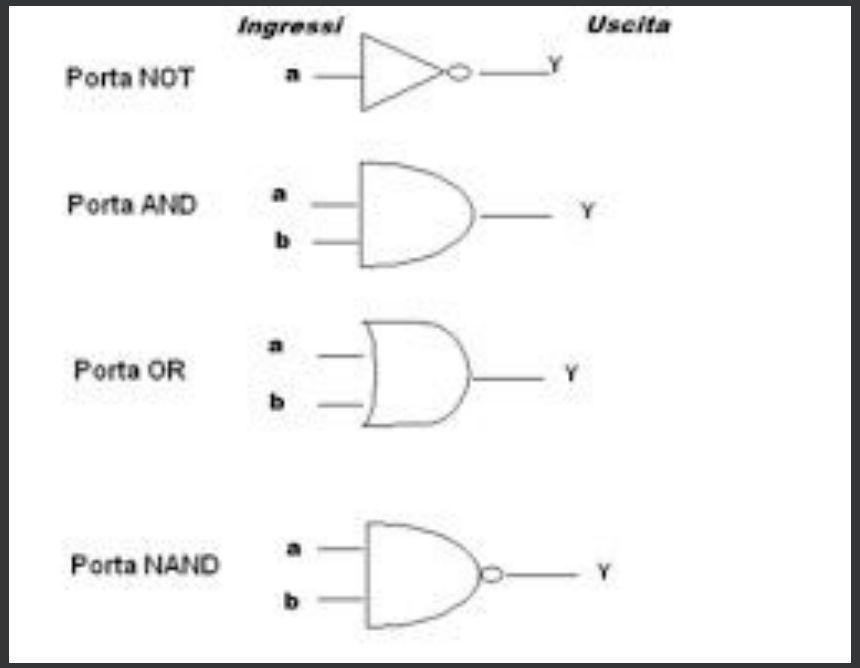


L'uscita Y dipende unicamente dall'ingresso X nell'istante in cui esso è presentato.

Abbiamo già visto degli esempi di macchine combinatorie?

Logiche Combinatorie

Esempi di Macchine Combinatorie sono le porte logiche di base di un circuito logico



Logiche Sequenziali

In una *logica sequenziale* l' output del sistema non dipende solo dall' input in un determinato istante, ma anche dalla sequenza di n (n dipenderà dalla macchina) input passati.

Detto in altri termini, a differenza delle macchine combinatorie, le **MACCHINE SEQUENZIALI** mantengono la memoria degli ingressi passati e reagiscono di conseguenza ad essi.

Macchine Sequenziali

- Come possiamo modellare il comportamento di una macchina sequenziale?
- Definire l'uscita date tutte le possibili sequenze di ingresso? IMPOSSIBILE! Il problema diventa ingestibile dopo poco tempo, soprattutto se l'uscita dipende da N istanti precedenti con N molto grande.
- Possiamo modellare il fatto che la macchina restituisca output diversi in base alla sequenza di N input passati dotando la macchina di uno STATO.
- Ogni qual volta la macchina riceve un input cambia il suo Stato: l'output all'istante successivo dipenderà sia dall'input in tale istante che dallo stato in cui si trova la macchina.
- Abbiamo trovato quindi una nuova definizione di macchina sequenziale, ovvero un sistema il cui output dipende dall'ingresso in un determinato istante (e solo da tale ingresso) e dallo STATO in cui si trova la macchina.

Macchina Sequenziale: Tabella di verità

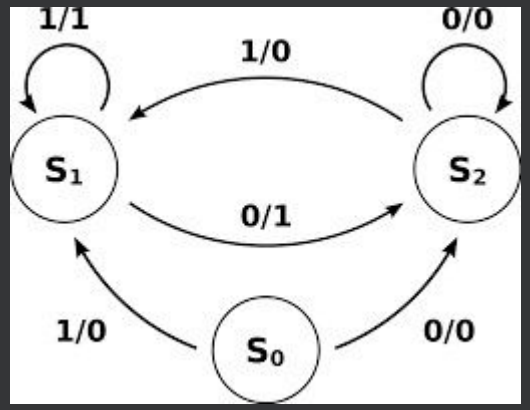
Un modo per rappresentare il comportamento di una macchina sequenziale è quello della **TABELLA DI VERITÀ**: vengono considerati in una tabella tutte le possibili combinazioni di ingressi e stati e in base ad essi lo stato successivo e l'output della macchina:

INPUT	STATO ATTUALE	STATO SUCCESSIVO	OUTPUT
I1	S1	S2	O1
I1	S2	S1	O2
I1	S3	S1	O2
I2	S1	S1	O5
I2	S2	S3	O4
I2	S3	S2	O2
...	...	...	...

Evidentemente questa modalità di rappresentazione risulta scomoda quando l'insieme dei possibili stati ed input è grande; si utilizza soprattutto in elettronica per rappresentare I circuiti logici più semplici.

Macchina Sequenziale: FSM

In informatica la modalità di rappresentazione delle macchine sequenziali è l' **FSM (Finite State Machine)** o Automa a Stati Finiti.

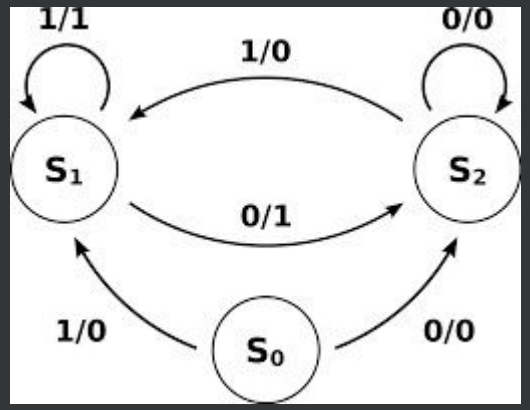


In tale rappresentazione utilizziamo un grafo, dove:

- I **nodi** rappresentano I possibili stati del sistema;
- Gli **archi** rappresentano le transizioni di stato e sono etichettati con due valori, l'input che provoca la transizione e il valore di uscita del sistema

Macchina Sequenziale: FSM

Vediamo la lettura di questa FSM



- Se il sistema si trova in S1 e riceve come input 1, resta in S1 e restituisce 1 in output;
- Sistema in S1, input 0: transizione in S2 e output =1;
- Sistema in S0, input 1: transizione in S1 e output=0;
- Ecc....

FSM ed AI

- Nel mondo dell'**AI**, molte delle tecniche di base per la modellazione dei comportamenti degli **Npc** prevede l'utilizzo delle FSM.

- Prima di vedere come possiamo modellare un AI attraverso un FSM, iniziamo ad immaginare come dovrebbe comportarsi un NPC "intelligente".

- Ovviamente il comportamento desiderato dipenderà fortemente dalla tipologia di gioco che stiamo considerando ma in generale:

    - Dovrà in qualche modo reagire ad una serie di Input dipendenti dal giocatore;

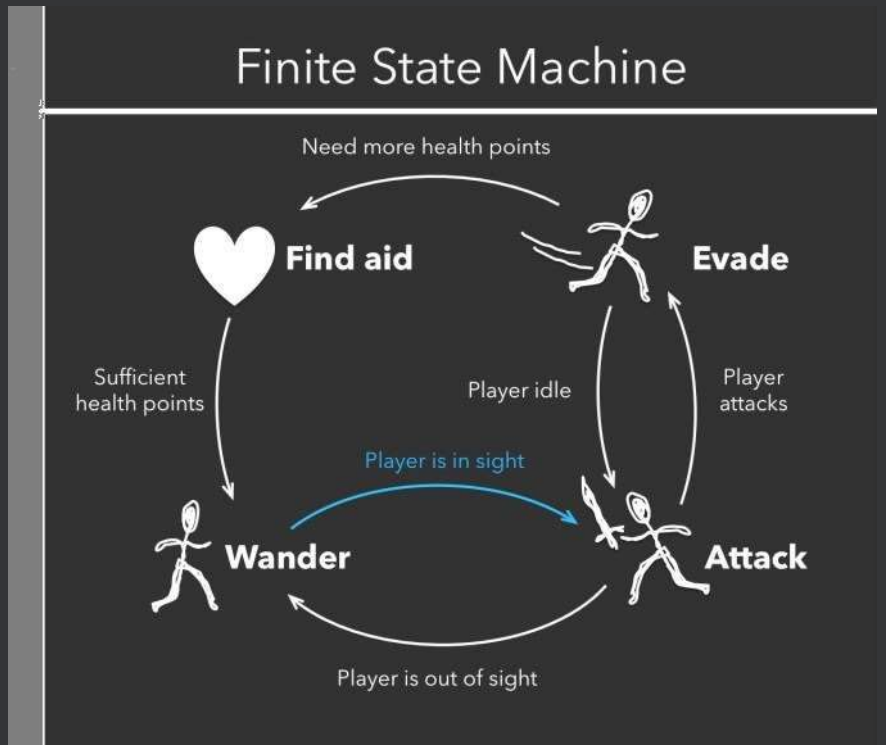
    - Il suo comportamento dovrà variare in base alla sua "percezione" del mondo di gioco in un determinato istante;

    - Dovrà effettuare una serie di azioni contestuali al suo stato attuale;

FSM ed AI

Un NPC si comporta in maniera diversa in base alla sua “percezione” del mondo di gioco.

Possiamo quindi immaginare di modellare il comportamento di un NPC con una FSM dove, il suo comportamento (insieme di azioni) in un dato momento rappresenta uno stato della FSM e dove le azioni del giocatore e l'evolversi del mondo circostante (in base al suo stato attuale) provocano delle transizioni nell’FSM.



AI: Caso di Studio

Comportamento dei soldati nello Stealth game Metal Gear Solid (Konami, 1998)

Metal Gear Solid è un gioco d’azione dove il protagonista deve cercare di infiltrarsi in una base nemica senza essere individuato dalle guardie.

Quali stati (o meglio macrostati) possiamo riconoscere come comportamento base dei nemici soldati?

AI: Caso di Studio

Comportamento dei soldati nello Stealth game Metal Gear Solid (Konami, 1998)

Metal Gear Solid è un gioco d'azione dove il protagonista deve cercare di infiltrarsi in una base nemica senza essere individuato dalle guardie.

Il comportamento di base dei soldati nemici si può suddividere in 4 Macrostati:

- Patrolling
- Suspect
- Active Chase
- Combat

Questa suddivisione è molto di macro-livello, sicuramente la macchina a stati reale implementata è ben più complessa. Ma questa visione d'insieme ci permette di capire molto bene come funzionano le AI implementate attraverso una FSM.

Patrolling

Il soldato è ignaro della presenza di Snake: continuerà a pattugliare la sua zona con un pattern predefinito

Suspect

Il soldato sospetta della presenza di Snake, inizierà un comportamento atto a “confermare” I propri sospetti:

Cosa lo fa scattare in tale stato?

- Ha “intravisto” Snake;
- Ha sentito un rumore;
- Ha individuato delle orme sulla neve;

Active Chase

Il soldato ha visto Snake e corre verso la sua posizione:

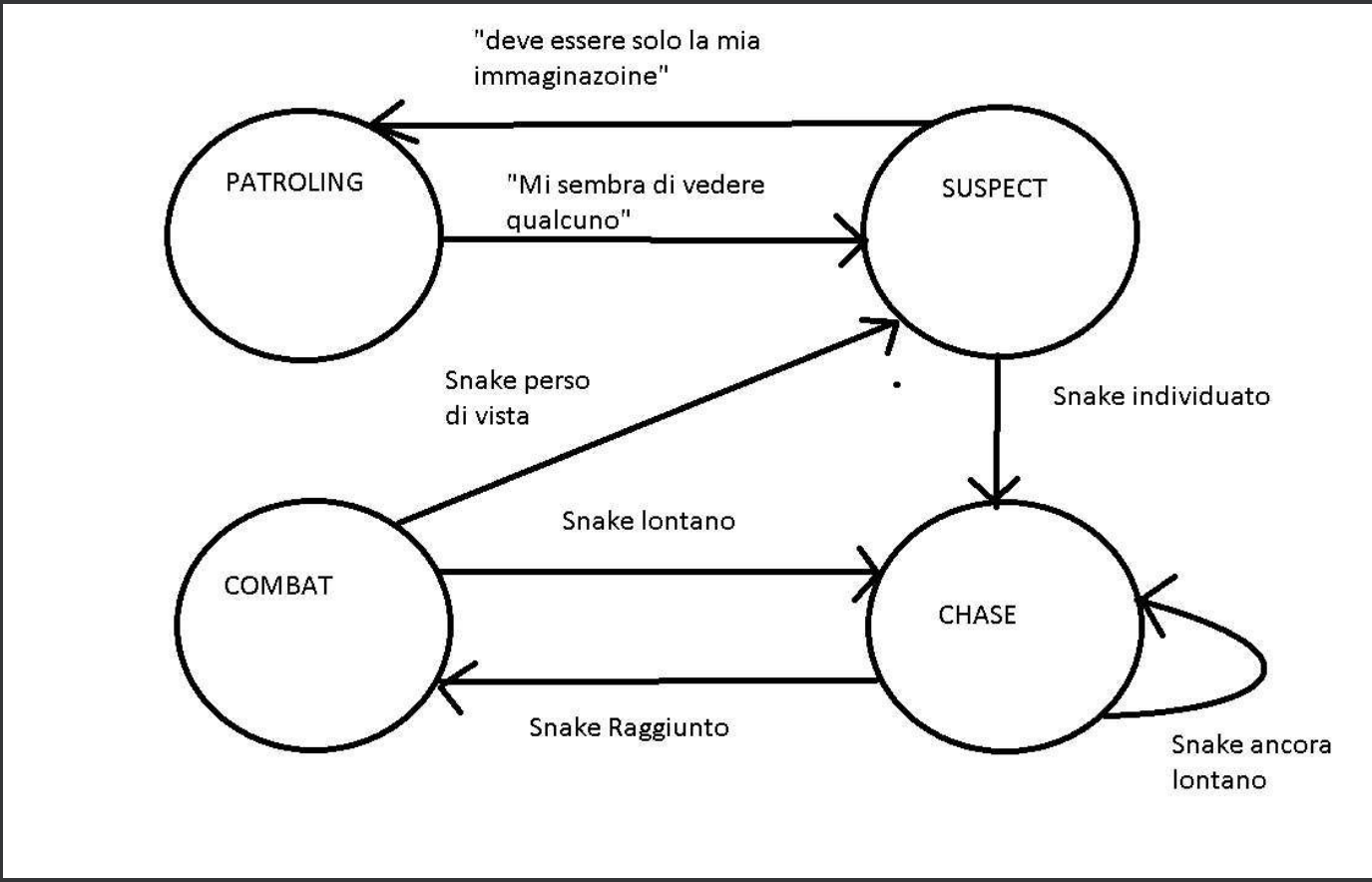
Combat

Il soldato è in modalità di combattimento e attuerà tutta una serie di sotto-comportamenti per avere la meglio sul nemico.

Tale stato può essere molto complesso(sub-fsm con molti stati o addirittura altre sub-fsm):

- Chiamare Rinforzi e usare oggetti di cura se si sta morendo;
- Ripararsi dietro ostacoli per evitare di essere colpiti
- Cambiare arma in base alla situazione
- ecc.

MGS: FSM


"deve essere solo la mia immaginazoine"
PATROLING
"Mi sembra di vedere qualcuno"
SUSPECT
Snake perso di vista
Snake individuato
Snake lontano
COMBAT
CHASE
Snake Raggiunto
Snake ancora lontano

AI ed FSM

In Definitiva, una volta modellato il comportamento di un NPC attraverso un FSM, risulterà poi semplice implementare tale comportamento.

Dovremo programmare una macchina a stati finiti per I nostri NPC dove definiremo il particolare comportamento per ogni possibile stato e le possibili transizioni tra gli stati.

Vediamo l'implementazione di una FSM (flessibile ed estendibile) all'interno del nostro engine Aiv.Fast2D.Component

Estendibilità e riutilizzo del codice

Partiamo dalla più semplice implementazione che possiamo fare di una FSM:

- State: classe base di uno stato (integrata nel game loop) che sarà estesa per rappresentare tutti gli stati del nostro gioco, e per ogni stato le transizioni verso altri stati

- FSM Component: componente che esegue la nostra macchina a stati (che non sarà altro che un insieme di stati).

Quali sono le problematiche di una implementazione del genere?

Una buona struttura

- ExecutableNode
- Action
- Condition
- State
- Transition
- StateMachine

Cosa manca?

Proviamo a discutere cosa manca!

Cosa manca?

1. Editor per costruire gli stati senza doverli costruire da codice (è molto noioso farlo da codice)
   a. Con la possibilità di definire degli assets che sono le FSM, così possiamo anche costruire template riutilizzabili (con il punto 2)

2. La possibilità di avere dei nodi nella FSM che sono delle FSM a tutti gli effetti. Quindi avere sotto-macchine a stati che vengono eseguite (in questo modo ad esempio potremmo non hardcodare tutto il patrol in un unico enorme stato, ma il patrol potrebbe essere semplicemente una sotto-macchina a stati).

Piccolo Esempio

Modellizziamo la AI di un semplice nemico di uno gioco 2D con visione dall'alto.

Questo nemico effettua pattuglia la zona passando da 4 punti. Se il player è abbastanza vicino inizia ad inseguirlo e quando è a una certa distanza si ferma e lo attacca.

Nello stato di attacco simuliamo l'attacco con un console.writeline ogni tot secondi dove scriviamo a console sparo ad un certo nemico.

Se il player si allontana torna ad inseguirlo, e se si allontana ancora ulteriormente torna a pattugliare la zona.