Máquinas de estado finitos

Una Máquinas de estado finitos (FSM, por sus siglas en inglés) es un modelo matemático compuesto por:

- Estados: Representan las distintas “modos” en los que puede estar el NPC (por ejemplo: patrullando, persiguiendo, atacando, huyendo).
- Transiciones: Reglas que definen cuándo y cómo el NPC cambia de un estado a otro (por ejemplo: si ve al jugador pasa de patrullando a persiguiendo).
- Eventos o condiciones: Son los disparadores de las transiciones (como la distancia al jugador, recibir daño, perder de vista al objetivo, etc.).
- Acciones: Lo que el NPC hace mientras está en un estado determinado (animaciones, movimientos, sonidos, etc.).

¿Cómo se modela?

Las FSM pueden implementarse utilizando simples estructuras condicionales if-then-else (o switch/case) es lo que se conoce como “FSM implícita”. Este tipo de FSM funcionan para casos simples, pero no escala bien y puede volverse difícil de mantener, depurar o extender.

Existen otras formas más robustas, modulares y limpias de implementar las FSM, uno de las técnicas más usadas es la que utiliza el modelo orientado a objetos donde cada estado es una clase, y las transiciones se manejan dentro de esas clases.

Las ventajas de este modelo es que cada estado encapsula su lógica lo que lo hace fácil de extender ya que añadir un nuevo estado solo requiere una nueva clase y vita grandes bloques de if/else que hacen el código ilegible.

Componentes de la FSM

class State:

Clase base para todos los estados del NPC. Cada estado define qué hacer al entrar, ejecutar y salir.

`def enter(self, npc):  → Acción al entrar en el estado.`

`def execute(self, npc):  → Acción principal que se ejecuta cada frame.`

`def exit(self, npc):  → Acción al salir del estado.`

`def get_next_state_class(self, event): → Devuelve la clase del estado siguiente según el evento recibido.`

class StateMachine

Esta clase implementa la Máquina de estados que gestiona el estado actual de un NPC.

Nuevos estados

Para definir nuevos estados se define una nueva clase que hereda de State:

```
class TimidoPatrolState(State):
    """
    Estado en el que el NPC tímido patrulla por el borde.
    """
...
```

Transiciones entre estados

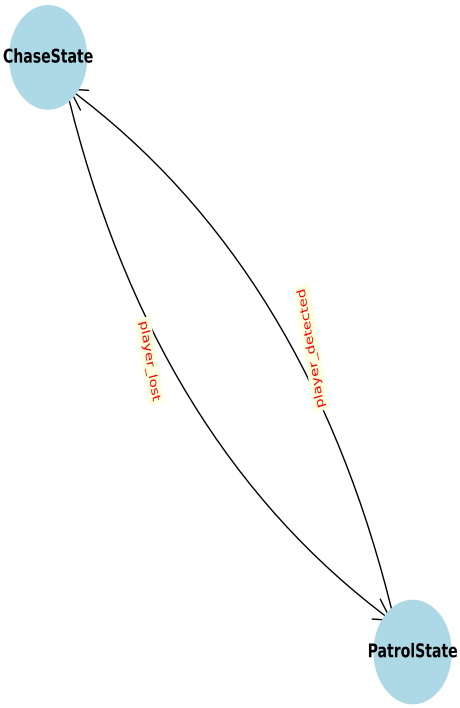
Hay que definir para cada estado un diccionario denominado transitions con el evento que provoca una transición y a qué estado se mueve.

```
# Definir transiciones para NPCTimido
TimidoPatrolState.transitions = {"player_seen": TimidoChaseState}
```

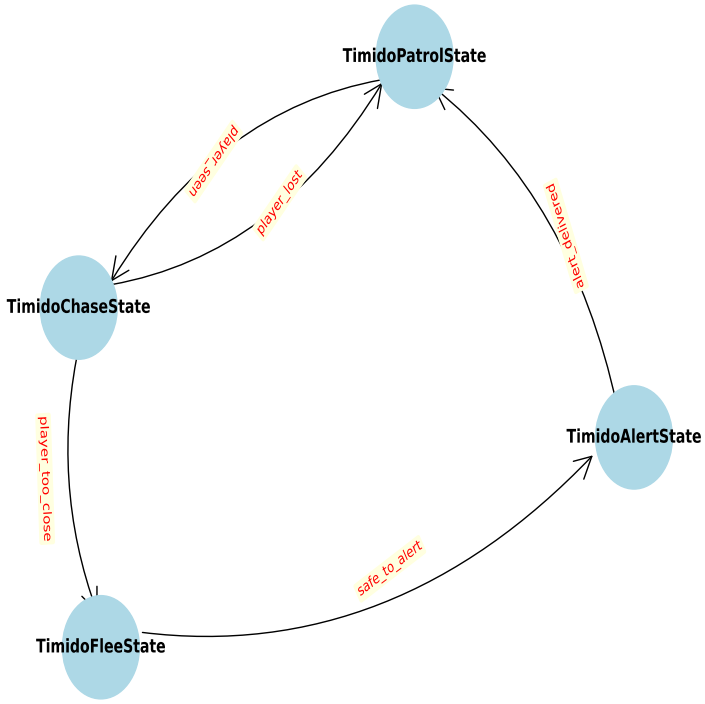
Ejemplo

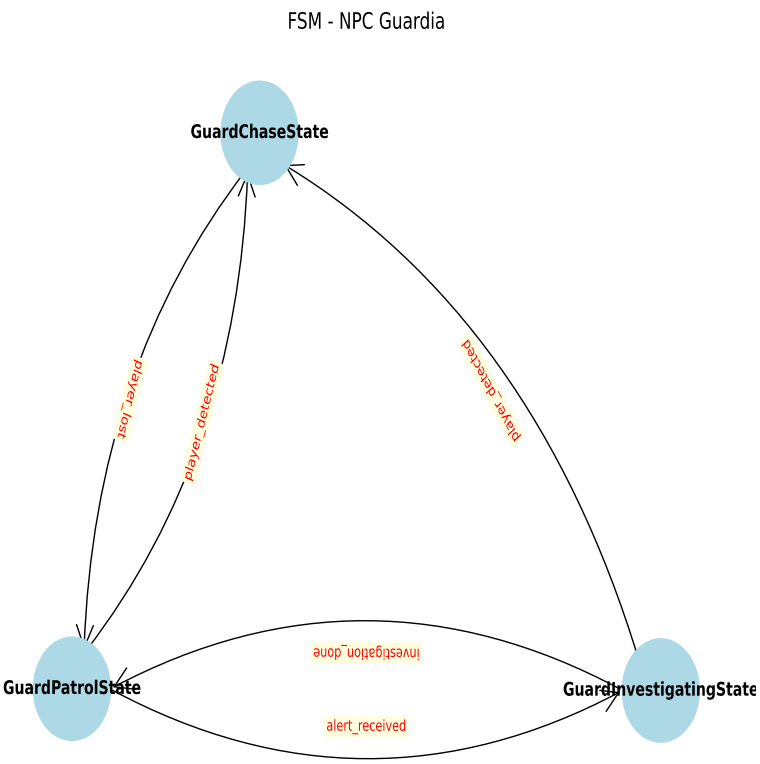
En esta imagen se muestra el comportamiento de tres NPCs distintos modelados usando FSM. Los estados se muestran en azul los eventos que provocan las transiciones entre estados aparecen sobre las flechas.

FSM - NPC Normal



FSM - NPC Tímido





Puedes ver y ejecutar el ejemplo en este [script de python]

Tracedump:

```
newBaseSize: 12pt
newBaseSizeInPt: 12
```