Máquina de Estados Finitos
Documentação

João Vitor Rossafa Teodoro
Victor Marquesini Portugal

Visão geral do cenário

O modelo desenvolvido se inspira num cenário de jogo de infiltração/stealth no qual o jogador deve evitar ser visto pelos inimigos. Tendo isso em mente, nossa máquina de estados simula dois agentes que vão estar vigiando o local com cada um seguindo sua rota. Ambos apresentam comportamentos simples, uma vez que o foco é na estruturação e organização do código.

O projeto segue o State Pattern, além de usar uma interface State e uma classe abstrata GenericState que recebem um tipo genérico para alcançar uma maior versatilidade com o código.

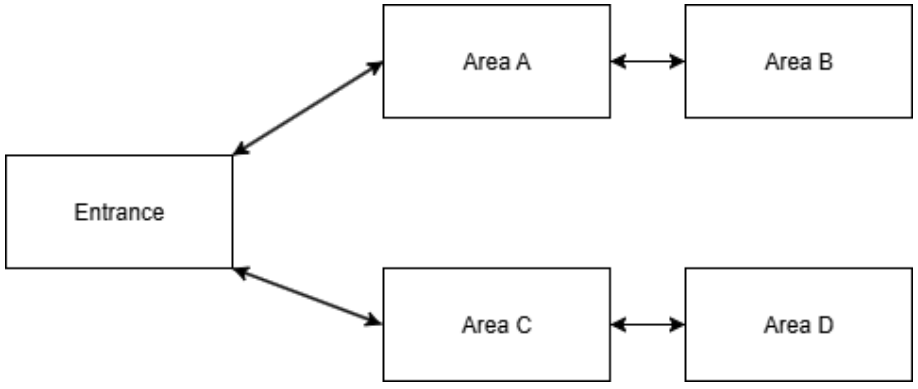
Visão geral de cada agente

Um deles é o PatrolGuard, que começa na entrada do local, vai até a Área A, então para a Área B, e depois para a Área A de volta antes de retornar para a entrada. Depois, ele vai para a Área C, Área D, Área C e retorna para a entrada. Então ele se repete.

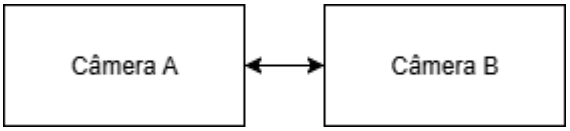
O CameraGuard é mais simples, seu estado se alterna entre estar olhando a Câmera A e a Câmera B, ficando nesse loop indefinidamente.

Diagrama de estados

PatrolGuard



CameraGuard



Regras por agente

PatrolGuard

Estado	Entrada	Execução	Saída	Transições
InspectingEntrance	n/a	Print “Inspecionando entra	n/a	InspectingAreaA, InspectingAreaC
InspectingAreaA	n/a	Print “Inspecionando Área	n/a	InspectingEntrance, InspectingAreaB
InspectingAreaB	n/a	Print “Inspecionando Área B”	print “Voltando para a entr	InspectingAreaA
InspectingAreaC	n/a	Print “Inspecionando Área		InspectingEntrance InspectingAreaD
InspectingAreaD	n/a	Print “Inspecionando Área	print “Voltando para a entr	InspectingAreaC

CameraGuard

Estado	Entrada	Execução	Saída	Transições
InspectingCameraA	Print “Mudando par câmera A”	Print “Inspecionando câmera A”	n/a	InspectingCameraB
InspectingCameraB	Print “Mudando par câmera A”	Print “Inspecionando câmera A”	n/a	InspectingCameraA

Descrição de variáveis por agente

PatrolGuard

isGoingToAreaB

- Tipo: Booleana
- Valor inicial: true
- Limiar para troca: se for true, o agente segue a rota A>B>A>Entrada, caso seja falsa, C>D>C>Entrada.
- Alteração: É alterada para false ao chegar na área B e para true ao chegar na área D

isReturningToEntrance

- Tipo: Booleana
- Valor inicial: false
- Limiar para troca: se for true, o agente vai para a entrada caso esteja nos estados InspectingAreaA ou InspectingAreaC. Caso contrário, o agente transiciona para InspectingAreaB ou InspectingAreaD a depender de qual branch está no momento.
- Alteração: É alterada para true ao chegar no final de uma branch (InspectingAreaB e InspectingAreaD) e para false uma vez que retorna para a entrada)

CameraGuard

cameraCheckCount

- Tipo: int
- Valor inicial: 0
- Limiar para troca: ao chegar em 3 o agente altera a câmera que está olhando.
- Alteração: Aumenta em 1 cada vez que o método execute() é chamado. Ao mudar de câmera, é resetado para 0.

Estrutura do código

Main (classe)

- Mantém referência dos dois agentes
- Chama o método update() de cada um para executar o estado atual

Character (interface)

- Indica os métodos que um personagem (agente da FSM) deve implementar
    - update()
    - updateState(AbstractState newState)
    - printStats(String currentStateMessage)

State (interface)

- Indica os métodos que um estado deve implementar:
    - enter()
    - execute()
    - leave()

AbstractState (classe abstrata)

- Mantém a referência do agente que possui os estados que herdam de AbstractState
- Implementação vazia dos métodos enter() e leave() da interface State, já que apenas o método execute() é de fato obrigatório

PatrolGuard

- Um dos agentes da FSM
- Seus estados são:
    - InspectingEntrance
    - InspectingAreaA
    - InspectingAreaB
    - InspectingAreaC
    - InspectingAreaD

CameraGuard

- Um dos agentes da FSM
- Seus estados são:
    - InspectingCameraA
    - InspectingCameraB

É importante destacar que, por seguir o State Pattern, cada estado citado é uma classe/arquivo no projeto. Além disso, ambos agentes são responsáveis por chamar o método execute() do estado ativo, e também de chamar o leave() do estado antigo e o enter() do novo ao mudar de estado.

Resultados esperados (logs)

PatrolGuard

-> Inspecionando entrada
-> Inspecionando área A
-> Inspecionando área B
   Voltando para entrada
-> Inspecionando área A
-> Inspecionando entrada
-> Inspecionando área C
-> Inspecionando área D
   Voltando para a entrada
-> Inspecionando área C
-> Inspecionando entrada

CameraGuard

-> Inspecionando câmera A
-> Inspecionando câmera B
-> Inspecionando câmera A
-> Inspecionando câmera B

Limitações

A maior falha é que os agentes sempre ficam engessados nesse loop e não existe nada que possa fazer eles quebrarem isso. Por exemplo, se o PatrolGuard for da entrada para a área A, ele sempre vai para a área B. Se houvesse um player (um terceiro agente) capaz de fazer barulho ou ser visto, ele poderia tirar o guarda dessa rota o que tornaria seu comportamento bem mais dinâmico.