DCC192

2025/2

UFMG

Desenvolvimento de Jogos Digitais

A20: IA — Modelagem de Comportamento

Prof. Lucas N. Ferreira

Plano de aula

- IA acadêmica vs. IA para jogos
- Comportamentos de NPCs
  - Fantasmas doPac-Man
- Máquinas de Estados Finita para Modelagem de Comportamento de NPCs
  - Estados
  - Transições

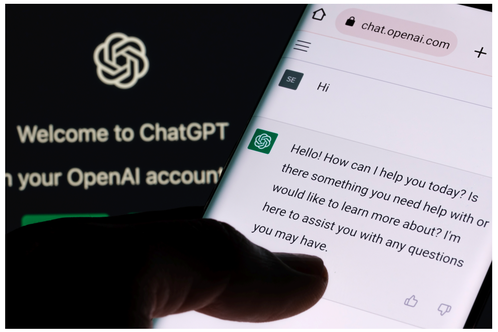
IA Acadêmica e Industrial

No contexto de Ciência da Computação, Inteligência Artificial (IA) é uma área de pesquisa interessada em simular a inteligência humana em sistemas computacionais:

Subáreas:

- Inferência Lógica e Probabilística
- Aprendizado de Máquina
- Processamento de Linguagem Natural
- Visão Computacional
- Robótica
- ...

Aplicações



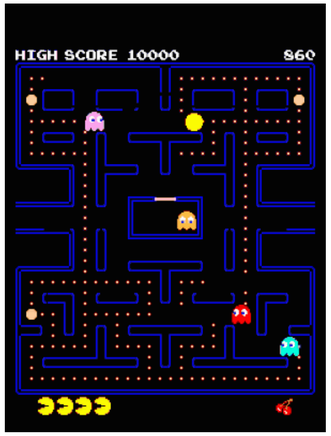
IA em Jogos: Tradicional

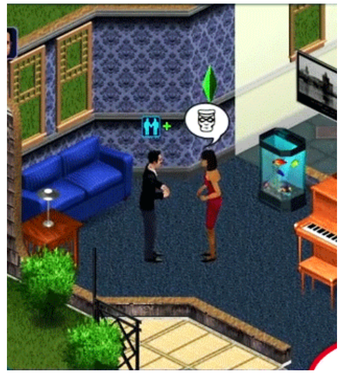
No contexto de Jogos Digitais, IA tradicionalmente se refere aos métodos utilizados para controlar objetos do jogo que não são controlados pelo jogador (NPCs):

Subáreas:

- Modelagem de Comportamento
- Estratégias Coletivas
- Encontrar Caminhos
- Steering Behavior
- Game Playing
- ...

Aplicações





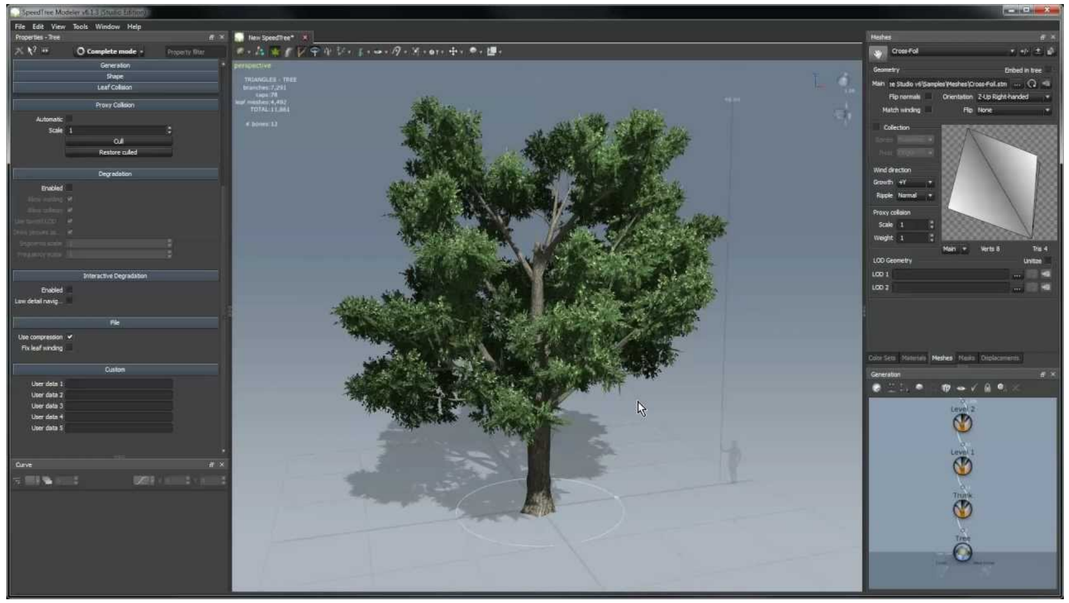
IA em Jogos: Emergente

Além do controle de personagens, IA vem cada vez mais sendo usada nos processos de desenvolvimento, produção e marketing dos jogos:

Subáreas:

- Geração Procedural de Conteúdo
- Modelagem de Usuário
- Matchmaking
- Churn Prediction
- ...

Aplicações



IA Acadêmica vs. IA para jogos

Objetivo

- Acadêmica: criar agentes cada vez mais inteligentes.
- Jogos: criar uma experiência de jogo divertida.

Recursos computacionais

- Acadêmica: recursos dedicados.
- Jogos: recusos compatilhados com outros subsistemas do jogo.

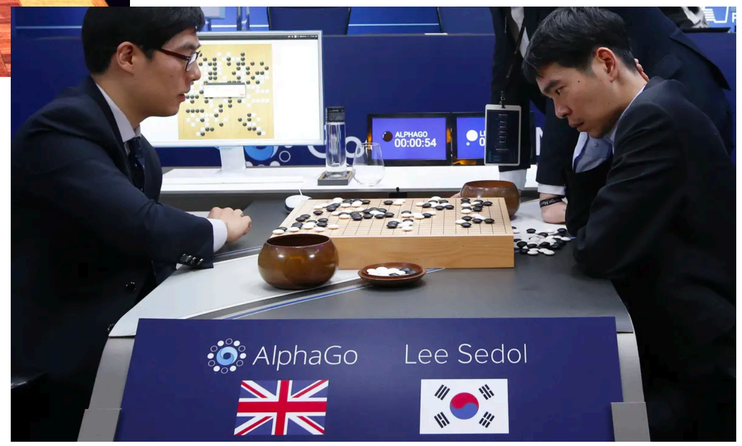
Jogos em Inteligência Artificial

A comunidade científica de IA tem um interesse histórico em jogos, pois eles são simulações simplificadas do mundo real, que para serem jogados, necessitam de uma série de habilidades:

- Reconhecimento de Padrões
- Planejamento
- Cooperação Social
- Pensamento estratégico
- ...

Deep Blue (1997)

Alpha Go (2016)



Exercício: Fantasmas do Pac-Man

Como você modelaria o comportamento dos fantasmas do Pac-Man?

...

Fantasmas do Pac-Man

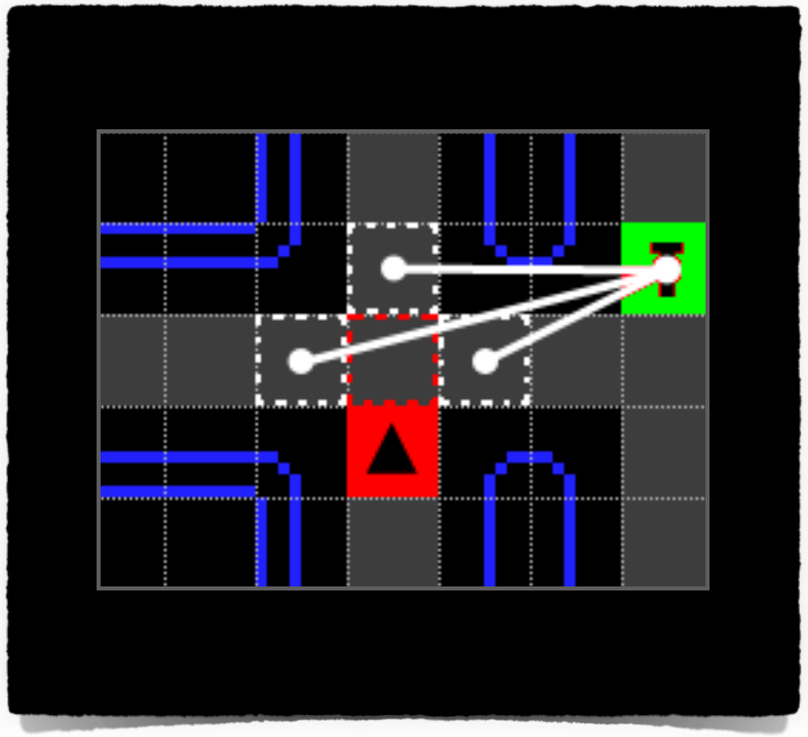
A intenção do designer do jogo era que cada fantasma tivesse uma “personalidade” específica de movimento, para que eles não ficassem apenas perseguindo o jogador constantemente:

“This is the heart of the game. I wanted each ghost to have a specific character and its own particular movements, so they weren’t all just chasing after Pac Man in single file, which would have been tiresome and flat.”

Toru Iwatani

Fantasmas do Pac-Man

O comportamento dos fantasmas consiste em se mover para uma **célula alvo** a partir da atual. Eles planejam seus caminhos em direção aos seus alvos da mesma maneira:



Planejamento uma célula de cada vez, sempre que entra em uma nova célula:

- *A próximo célula é escolhida dentre as vizinhos da atual.*
- *A célula vizinha com menor distância eucliana ao alvo **T** é escolhida.*
- *Nunca inverte a direção do seu movimento (não volta para trás)*

As diferentes personalidades surgem da maneira individual que cada fantasma seleciona seu vértice alvo.

Estados

Os fantasmas possuem quatro estados diferentes, que definem um ponto de referência para suas escolha de célula alvo:

- **Dispersão (scatter)**

  A célula alvo é um ponto pré-definido no labirinto

- **Perseguição (chase)**

  A célula alvo é definida em relação à posição do Pac-Man

- **Assustado (frightened)**

  Não existe célula alvo, eles se movem aleatóriamente no labirinto

- **Morto (dead)**

  A célula alvo é a casa dos fantasmas no centro do labirinto

Transições de Estados

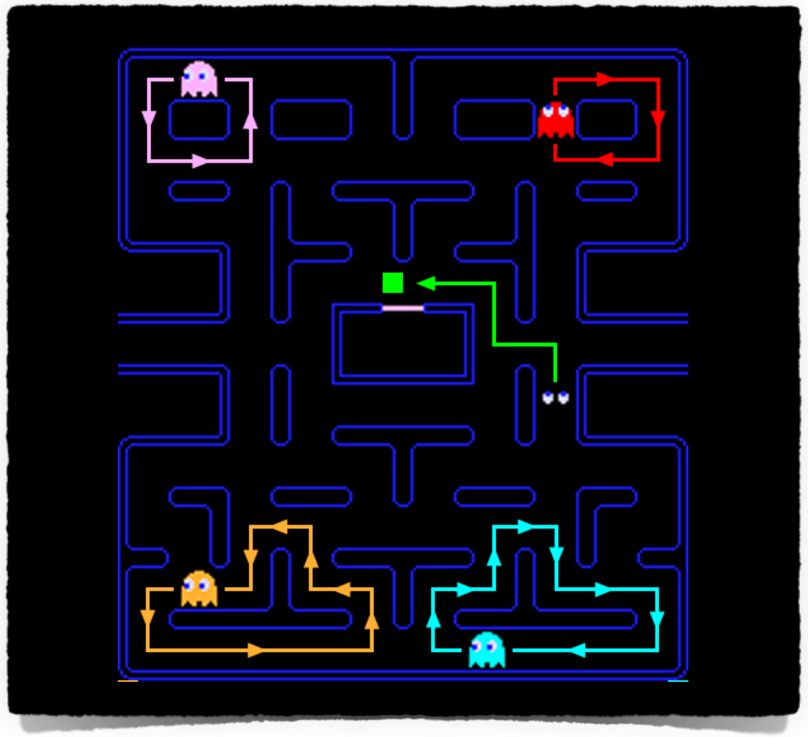
Em todas as fases, os fantasmas executam quatro ondas de **dispersão** alternada com **perseguição**. Ao final da quata onda, eles ficam no estado de **dispersão** até o final da fase.

A duração de cada estado varia ao longo das fases para criar uma progressão de dificuldade. Na primeira fase:

1. **Dispersão** por *7s*, depois **Perseguição** por *20s*
2. **Dispersão** por *7s*, depois **Perseguição** por *20s*
3. **Dispersão** por *5s*, depois **Perseguição** por *20s*
4. **Dispersão** por *5s*, depois **Perseguição** até o fim

Dispersão

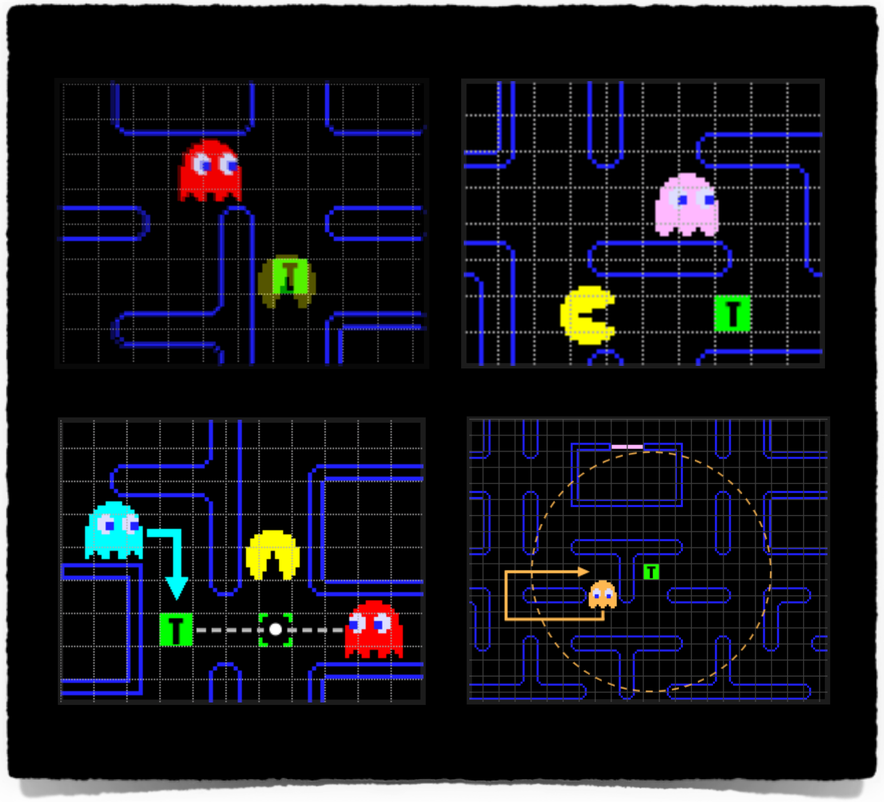
A célula alvo é um ponto pré-definido no labirinto. Cada fantasma tem uma célula pré-denida diferente (nos quatro cantos do labirinto):



- **Blinky (vermelho)**: canto direito superior
- **Pinky (rosa)**: canto esquerdo superior
- **Inky (ciano)**: canto esquerdo inferior
- **Clyde (laranja):** canto direito inferior

Perseguição

Cada fantasma usa uma estratégia diferente de perseguição:



- **Blinky (vermelho)**

  Persegue a último célula que o Pac-Man visitou

- **Pinky (rosa)**

  Persegue uma célula à frente da posição atual Pac-Man visitou

- **Inky (ciano)**

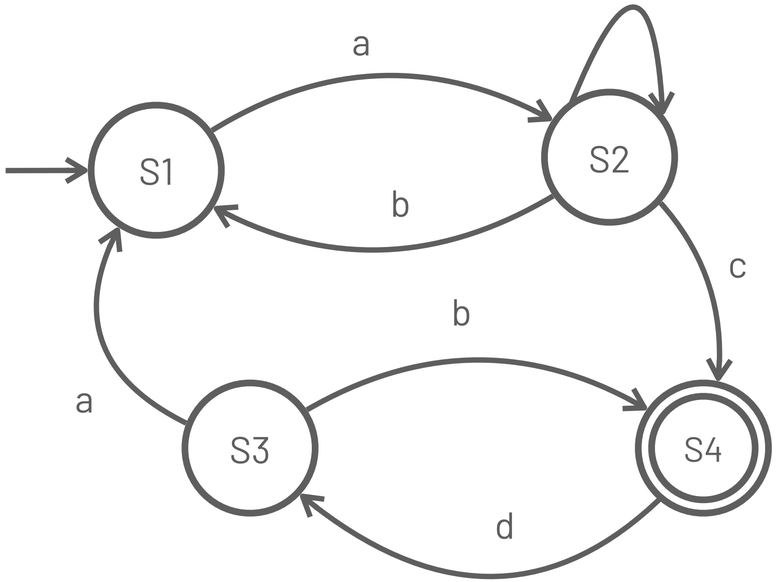
  Comporamento complexo que depende da posição do Blinky e do Pac-Man

- **Clyde (laranja)**

  Depedendo da distância euclidiana até o Pac-Man, Clyde persegue como Blinky ou volta para seu ponto de dispersão

Máquina de Estados Finita (FSM)

Uma **Máquina de Estados Finita** (FSM - *Finite State Machine*) é um modelo matemático tradicionalmente utilizado para representar programas de computador ou circuitos lógicos.



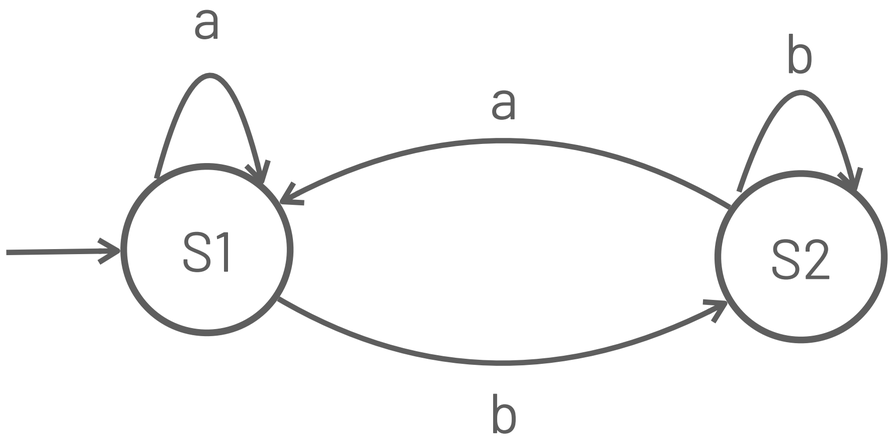
Uma FSM é definida por dois conjuntos:

1. **Estados** que a máquina pode estar (um por vez)
   S1, S2, S3, S4 (final)

2. **Condições** para transições de estados
   a, b, c, d

Exemplo: Catraca

Uma catraca é um exemplo muito simples de máquina de estados finita:



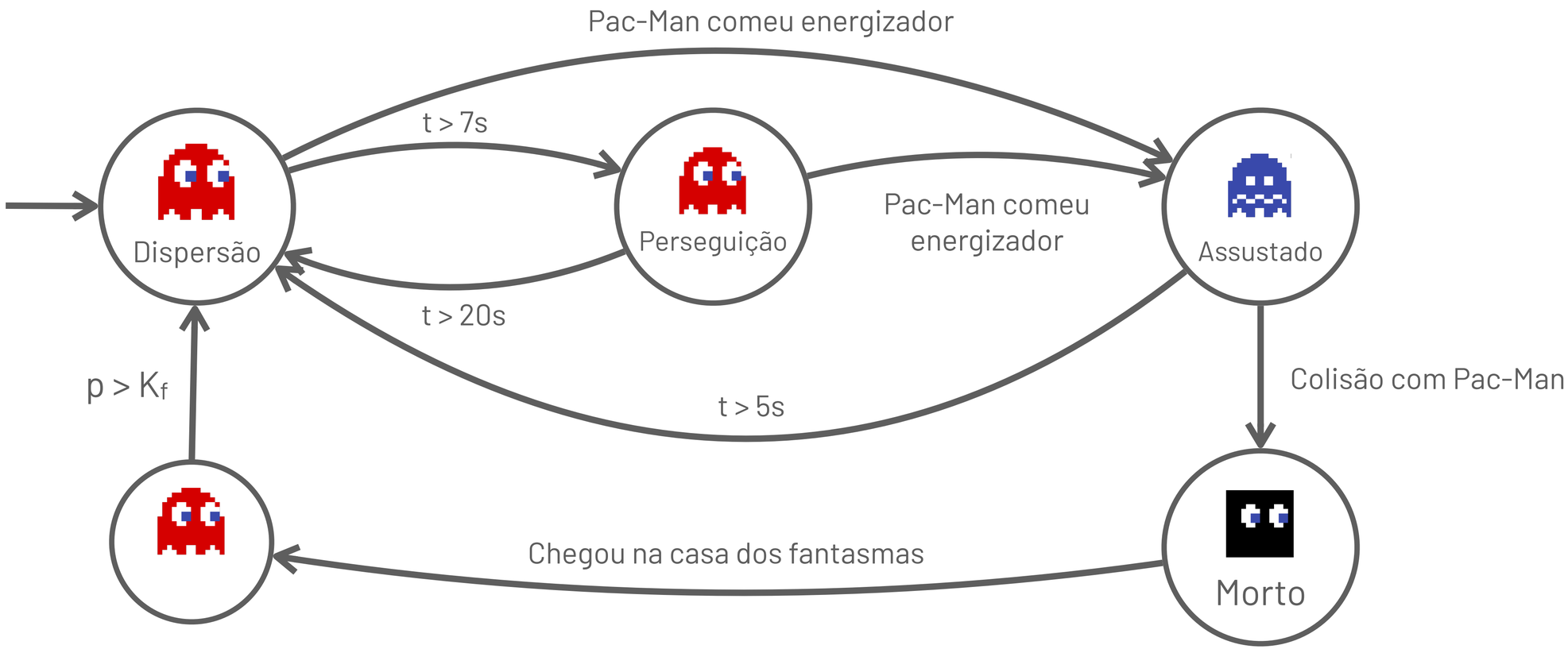
(S1) Trancada

- a: puxar a catraca
- b: encostar o cartão

(S2) Destrancada

- a: puxar a catraca
- b: encostar o cartão

FSM dos Fantasmas do Pac-Man



- t: tempo em segundos desde o início do estado
- p: número de pastilhas comidas pelo Pac-Man
- K_f: número de pastilhas que o Pac-Man precisa comer para que o fantasma f entre no labirinto

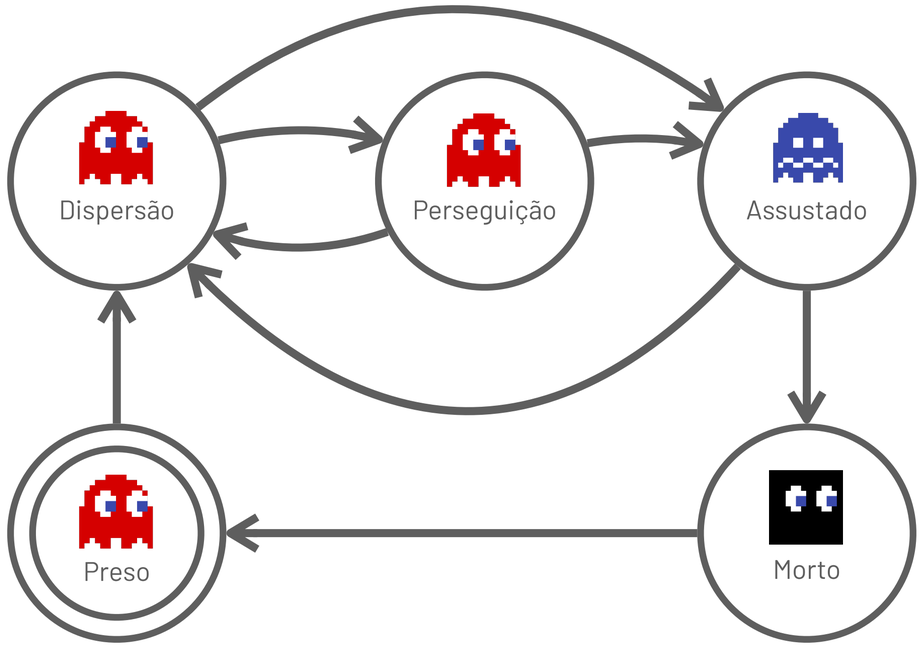
Implementando FSMs

As técnicas mais comuns para implementação de FSMs são:

- Comando Switch/Case
- Padrão de Projeto State
- Interpretadores (mais usado para IA dos personagens)

Implementação de FSMs: Interpretadores

Todas as transições são codificadas em um só lugar, escritas como uma grande verificação condicional com múltiplas ramificações (instruções case em C++).



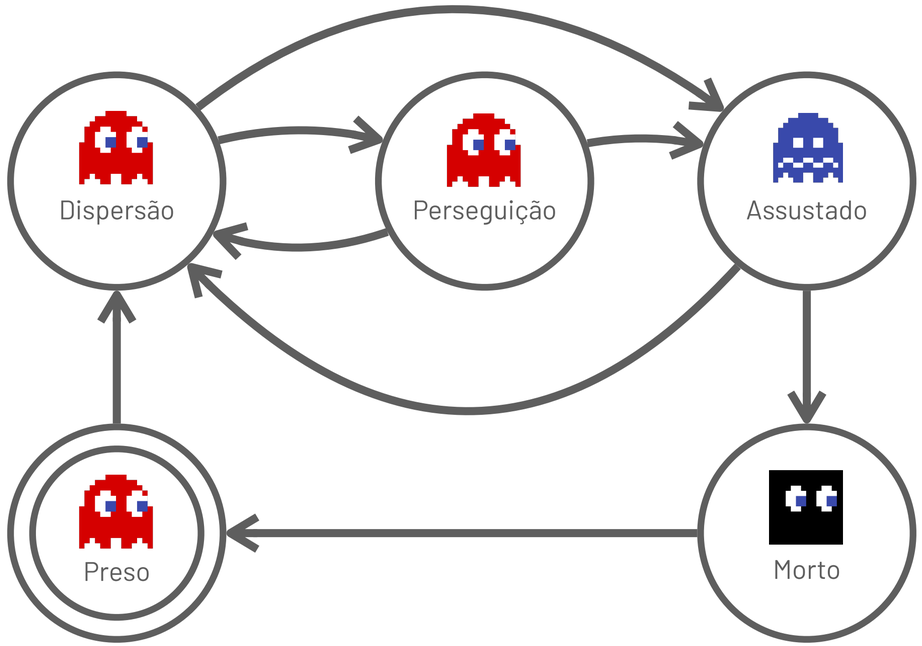

```
enum State {
  STATE_SCATTER,
  STATE_CHASE,
  STATE_FRIGHTED,
  STATE_DEAD,
  STATE_LOCKED
};

switch (state) {
    case STATE_SCATTER:
      ...
      break;

    case STATE_CHASE:
      ...
      break;
     ...
}
```

Implementação de FSMs: Padrão de Projeto State

Cada estado é responsável por determinar seu estado sucessor, portanto a lógica de transição é separada em vários pequenos pedaços.




```
class GhostState {
public:
    virtual GhostState() {}
    virtual void enter(Input i) {}
    virtual void update(float dt) {}
    virtual void exit() {}
};
```

```
class ScatterState : public GhostState {
public:
    void enter(Input i) { ... }
    void update(float dt) { ... }
    void exit() { ... }
}
```

Implementação de FSMs: Interpretadores

Um interpretador FSM normalmente é escrito usando uma abordagem muito semelhante ao padrão de estado, porém os estado e suas transições são definidas em um arquivo externo.

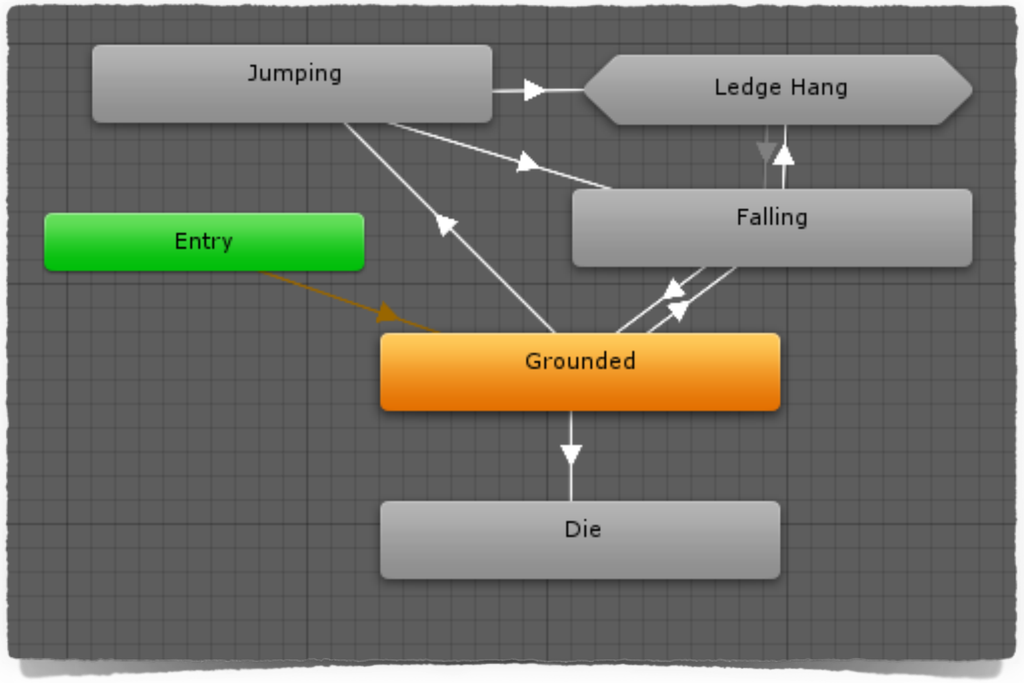


Figura 1: FSM da Unity Engine

Próxima aula

A21: Pathfinding

- Representação de Mapas em Jogos
- Busca em Largura
- Algoritmo de Djikstra
- Algoritmo A* e Heurísticas