INF216

2023/2

UFV

Projeto e Implementação de Jogos Digitais

A8: IA - Máquina de Estados Finita

Logística

Avisos

- Entrega do P4: Super Mario Bros hoje às 23:59!
- Entrega do PF: Documento de Design sexta-feira às 07:30

Última aula

- Câmeras 2D

Plano de Aula

- Inteligência Artificial
  - Ciência da Computação
  - Jogos Digitais
- Máquina de estados finita
- Pacman
  - Modelagem dos fantasmas

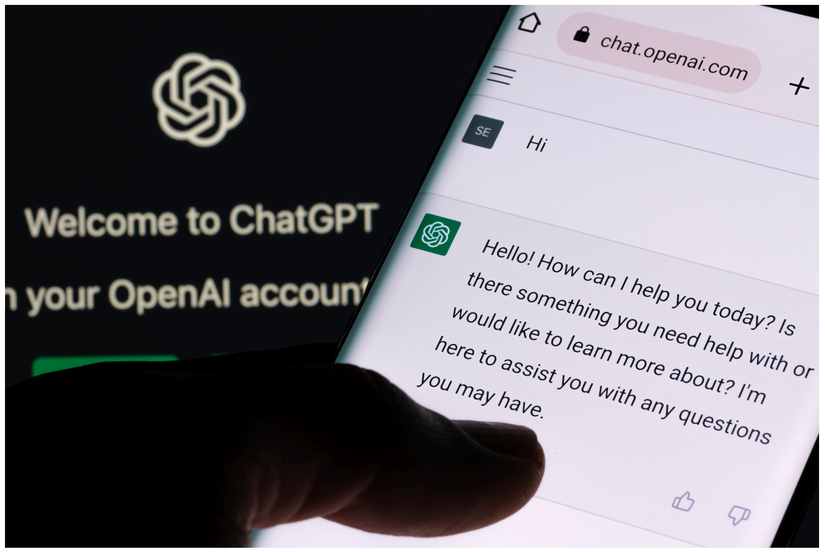
Inteligência Artificial

Em Ciência da Computação, I**nteligência artificial (IA)** é uma área de pesquisa interessada em simular a inteligência humana em sistemas computacionais:

- Resolução de problemas
- Representação de conhecimento, raciocínio e planejamento
- Aprendizado
- Comunicação e percepção

...

Inteligência Artificial



Inteligência Artificial em Jogos

Tradicional

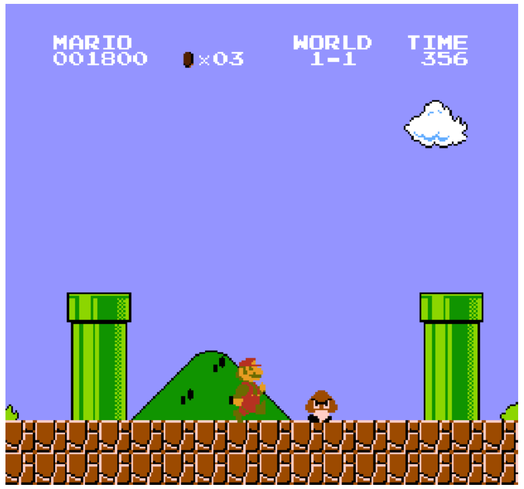
Em jogos digitais, IA tradicionalmente se refere aos métodos utilizados para controlar objetos do jogo que não são controlados pelo jogador:

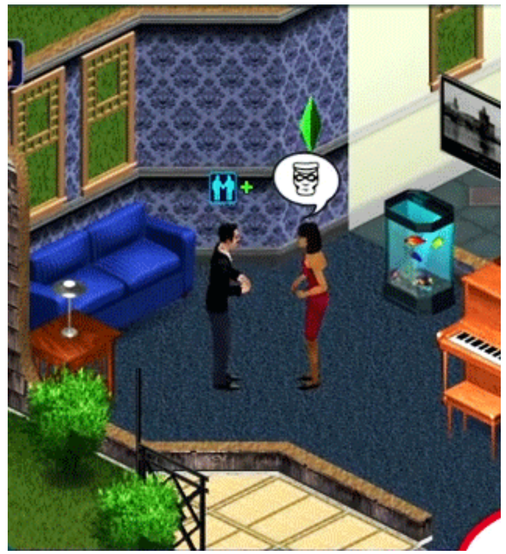
- Inimigos individuais
- Personagens não jogáveis (NPCs)
- Carros de corrida
- Estratégias de batalha (inimigos coletivos)

...

Inteligência Artificial em Jogos

Tradicional





Inteligência Artificial em Jogos

Emergente

Além do controle de personagens, IA vem cada vez mais sendo usada em processos de produção de jogos.

- Geração procedural de conteúdo durante o desenvolvimento
- Modelagem de usuário para marketing direcionado
- Associação (match) entre jogadores de mesmo nível para balanceamento

...

IA vs. IA para Jogos

Objetivo

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

Recursos computacionais

- Computação: 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:

- Processamento de cores, números, formatos, etc.
- Associação de padrões
- Planejamento
- Cooperação Social
- Pensamento estratégico

Jogos em Inteligência Artificial

Deep Blue (1997)

Alpha Go (2016)

Métodos Tradicionais de IA para Jogos

Uma grande quantidade de comportamentos diferentes pode ser modelada em jogos com três seguintes métodos:

- Máquina de estados finita
- Busca de caminhos (*pathfinding*)
- Comportamentos de navegação (*steering behaviors*)

Máquina de Estados Finita

Fantasmas do Pac-Man

Como você implementaria os fantasmas do Pac-Man?

Pac-Man

Intenção Artística

"All the computer games available at the time were of the violent type - war games and space invaders. There were no games that everyone could enjoy. I wanted to come up with a "comical" game everyone could enjoy." - Toru Iwatani

Público-alvo

Qualquer pessoa

Objetivo do jogo

Comer todas as pastilhas enquanto foge dos 4 fantasmas que assombram o labirinto.

Regras

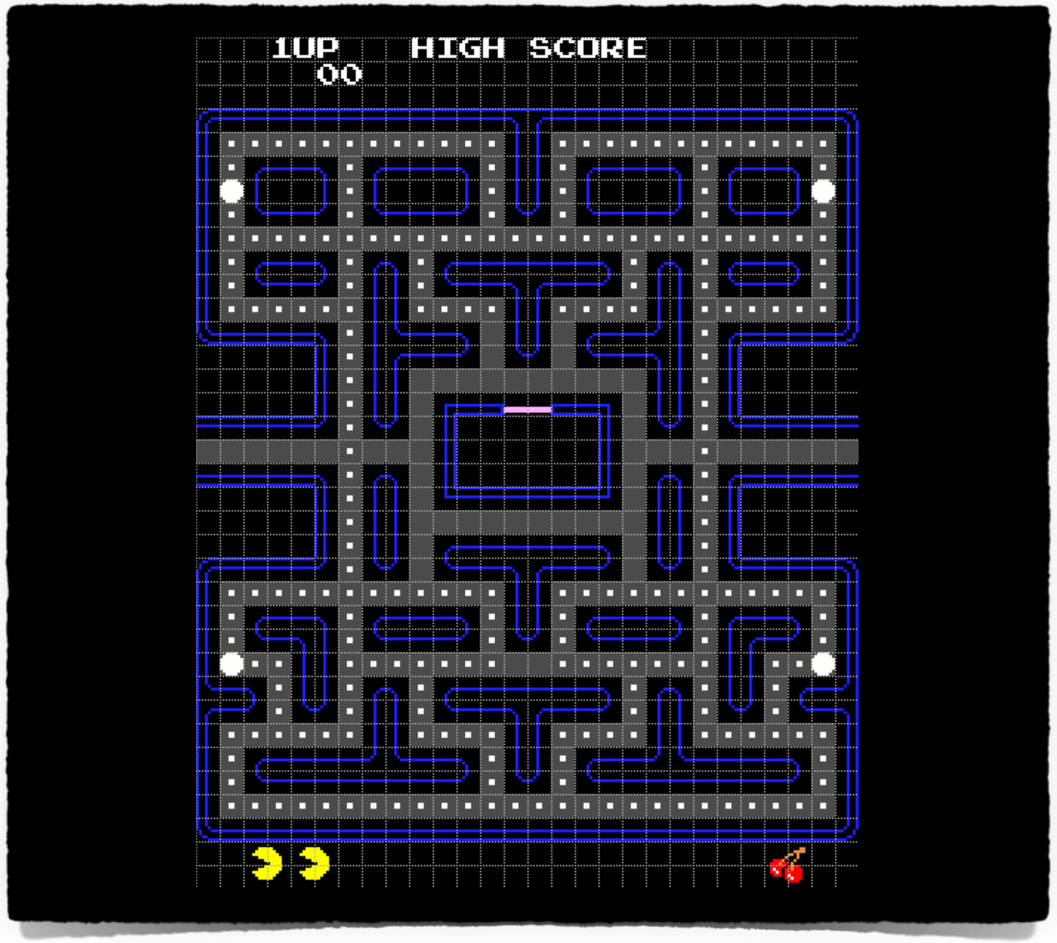
Se o Pac-Man colidir com um fantasma:

1. Ele perde uma vida e as posições dele e dos fantasmas são redefinidas para seus locais iniciais
2. As pastilhas consumidas permanecem dessa forma

Se o Pac-Man colidir com uma pastilha energizadora:

1. Os fantasmas ficam assustados e fogem por um curto período de tempo
2. O Pac-man pode comer fantasmas para ganhar pontos extras durante este período
3. Os fantasmas comidos retornam à sua posição inicial e após algum tempo voltam para o labirinto

Sistema de Progressão



1. O labirinto é o mesmo em todas fases, com 240 pastilhas comuns e 4 energizadoras

2. A dificuldade das fases aumenta por meio da alteração da velocidade do Pac-Man e do comportamento e velocidade dos fantasmas

Fantasmas

"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

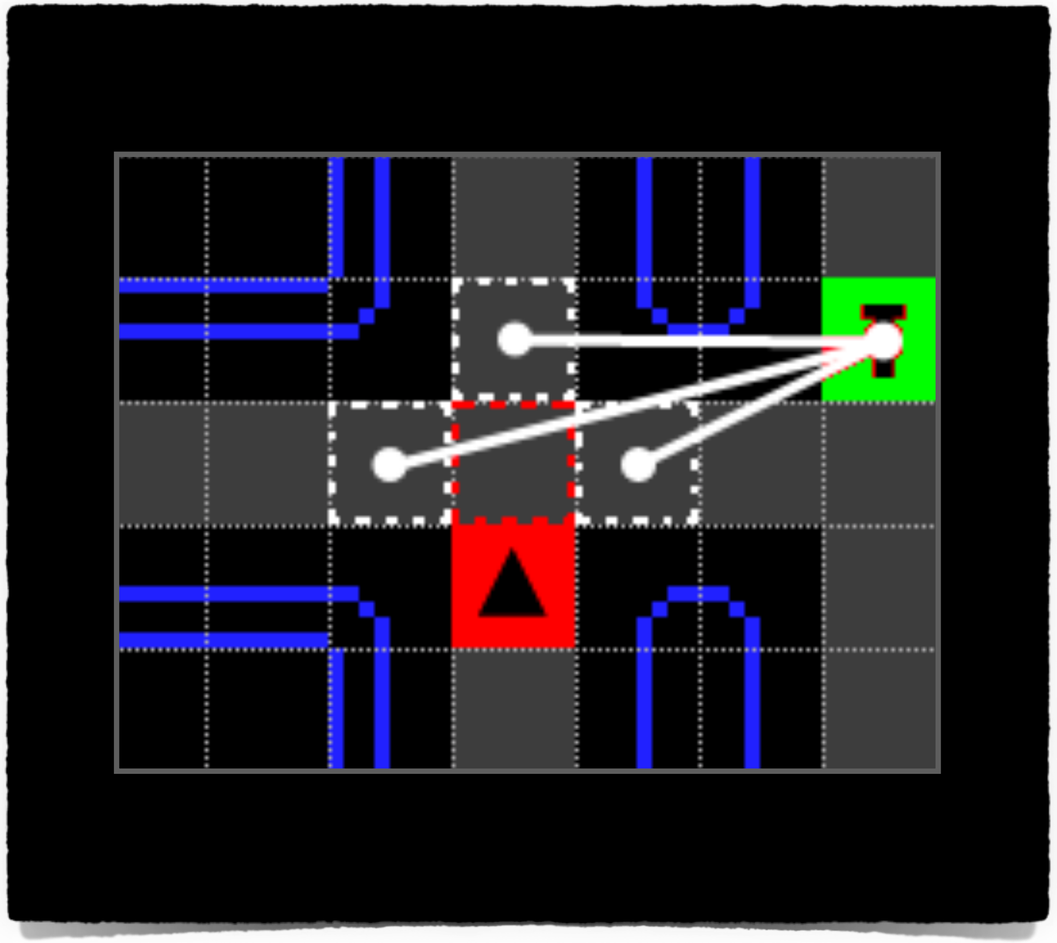
Fantasmas

O comportamento dos fantasma consiste em se mover para um **vértice (célula) alvo** a partir do atual.

Os 4 fantasmas planejam seus caminhos em direção aos seus alvos da mesma maneira.

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

Planejamento de Caminho



Os fantasmas planejam seu caminho em direção ao alvo (**T**) um passo de cada vez.

Sempre que um fantasma entra em uma novo vértice, ele decide o próximo vértice dentre os vizinhos do atual.

O vértice vizinho com menor distância eucliana ao alvo **T** é escolhido.

Um fantasma nunca inverte a direção do seu movimento (voltar para trás).

Estados

Os fantasmas possuem 4 estados diferentes, que definem um ponto de referência para suas escolha de vértice alvo.

- **Dispersão (scatter)**

  O vértice alvo é um ponto pré-definido no labirinto.

- **Perseguição (chase)**

  O vértice alvo dos fantasmas é definido em relação à posição do Pac-Man

- **Assustado (frightened)**

  Não existe um vértice alvo, os fantasmas se movem aleatóriamente no labirinto.

- **Morto (dead)**

  O vértice alvo é a casa dos fantasmas no centro do labirinto.

Transição de Estados

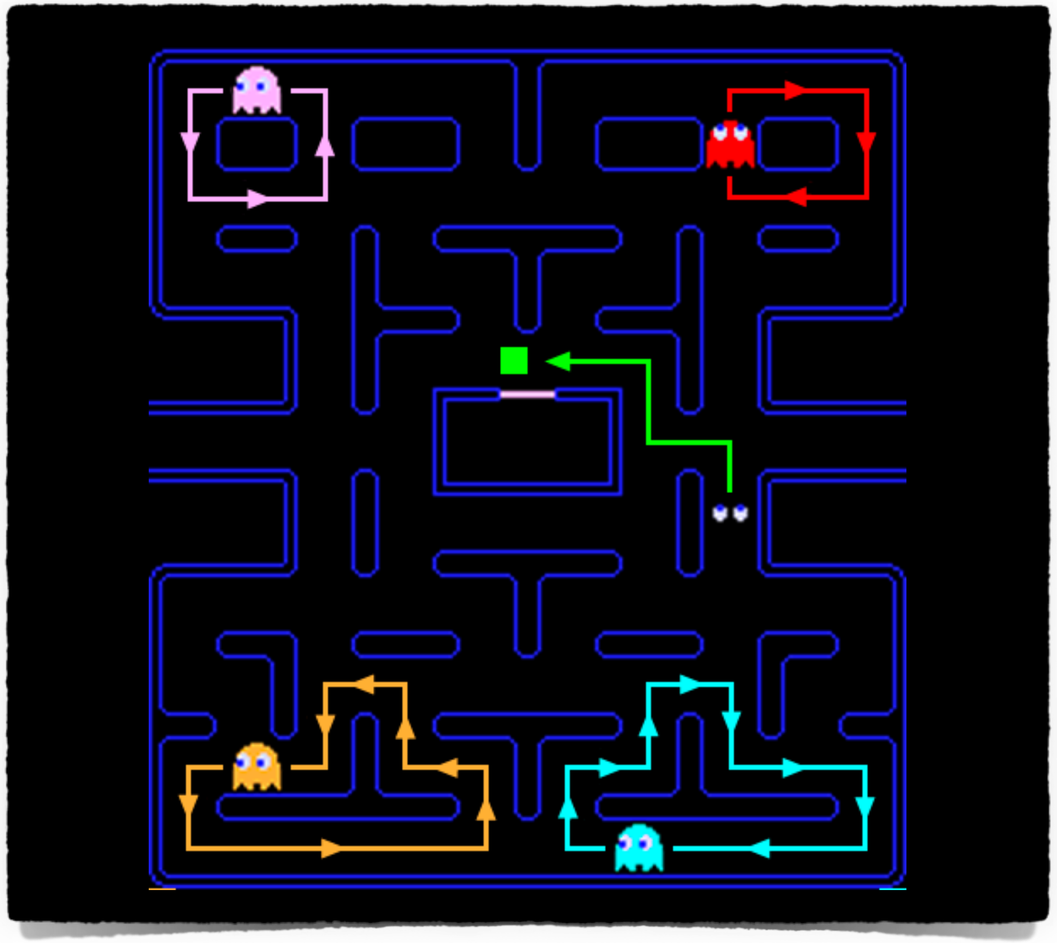
Em todas as fases, os fantasmas executam 4 ondas de **dispersão** alternada com **perseguição**.

Ao final da quata onda, os fantasmas ficam no estado de **dispersão** até o final da fase (vitória ou derrota).

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

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

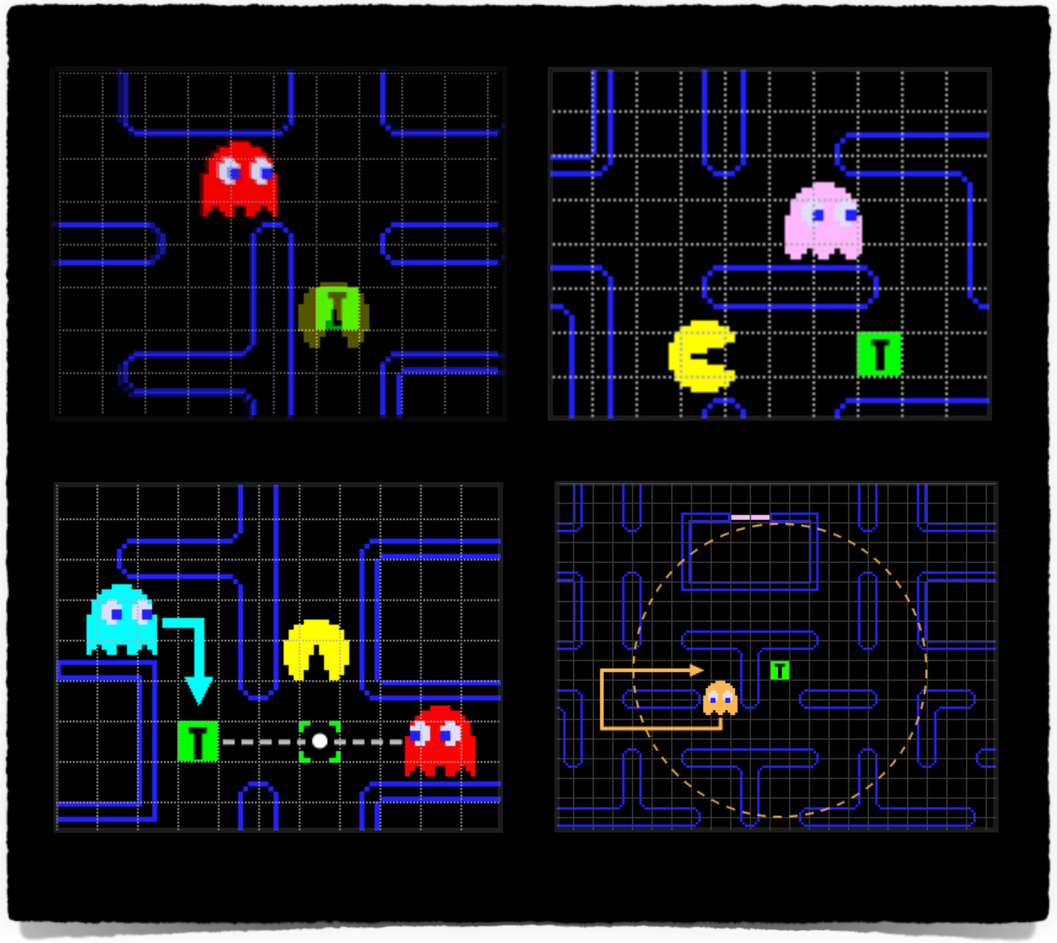
Dispersão



O vértice alvo é um ponto pré-definido no labirinto.

Cada fantasma tem uma posição pré-denida diferente (nos 4 cantos do labirinto).

Perseguição



Cada fantasma usa uma estratégia diferente de perseguição:

- **Blinky (vermelho)**

  Persegue o último vértice que o Pac-Man visitou.

- **Pinky (rosa)**

  Persegue um vértice à frente da posição atual Pac-Man visitou.

- **Inky (ciano)**

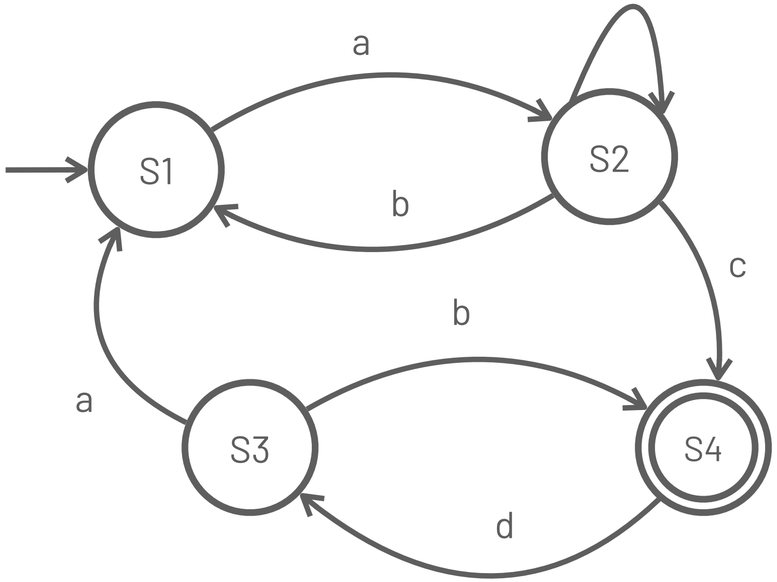
  Comporamento complexo que depende da posição do Blinky e do Pac-Man.

- **Clyde (laranja)**

  Depedendo da distância euclidiana até o Pac-Man, Clyde persegue o Pac-Man 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.

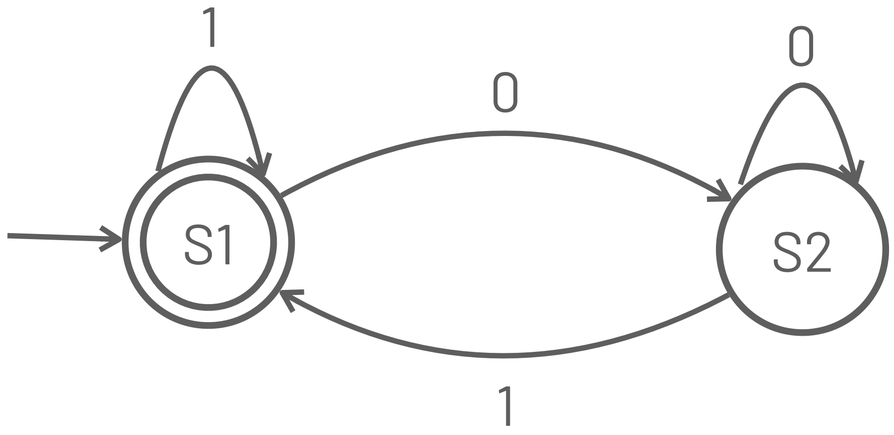


Uma FSM é definida por dois conjuntos:

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



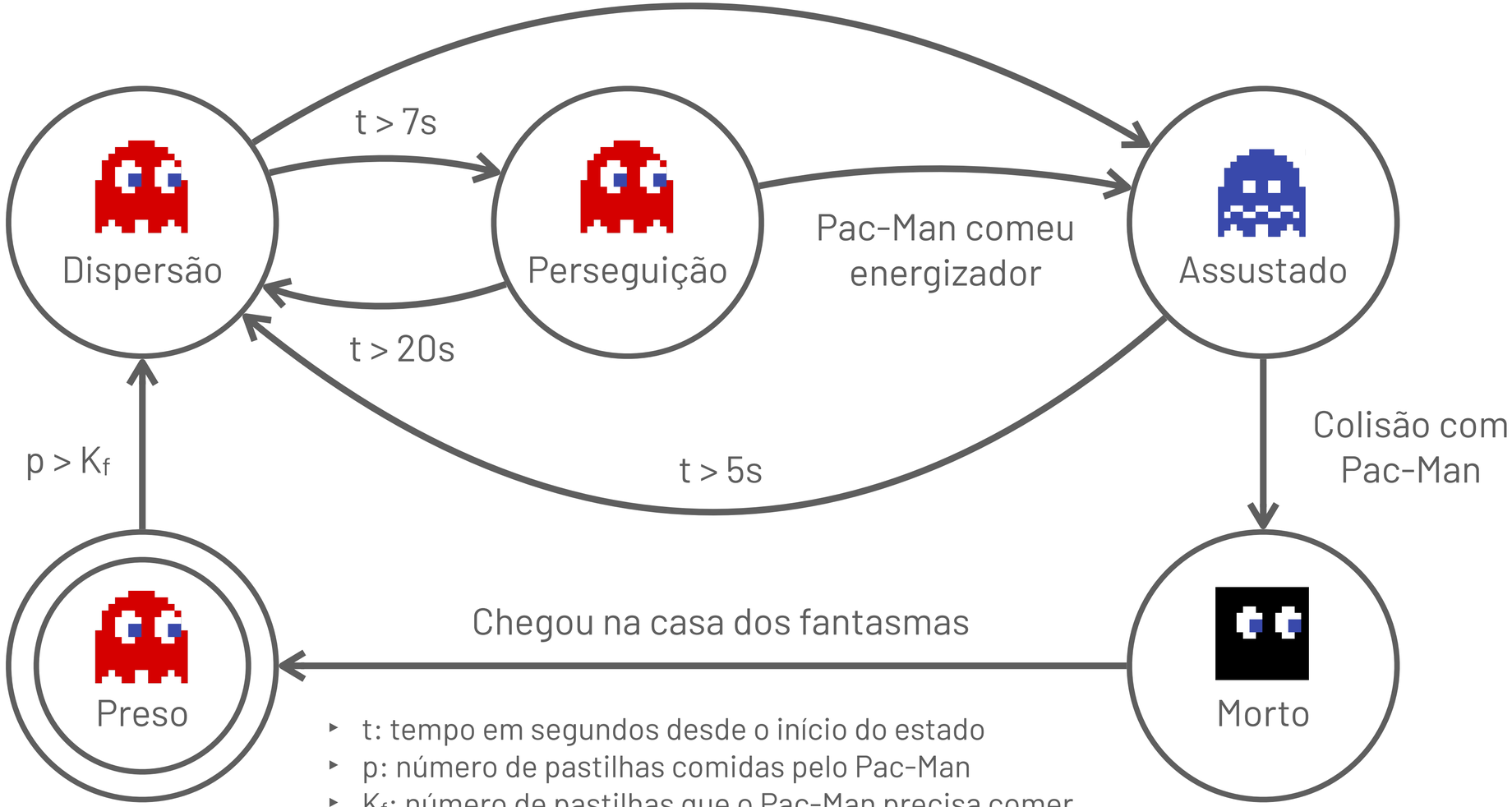
(S1) Trancada

- 0: inserir uma moeda
- 1: puxar a catraca

(S2) Destrancada

- 0: inserir uma moeda
- 1: puxar a catraca

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

Implementação de FSMs

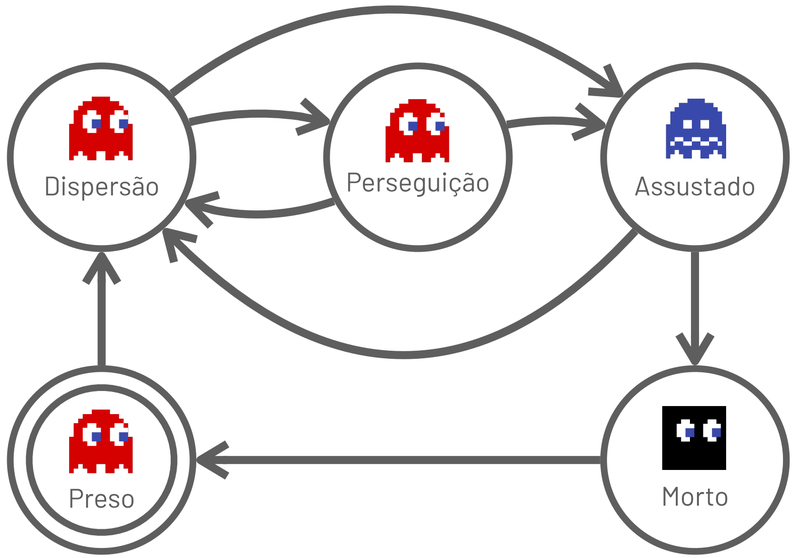
Técnicas mais comuns para implementação de FSMs:

- Comando Switch/Case
- Padrão de Projeto State
- Interpretadores

Implementação de FSMs

Comando Switch/Case

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

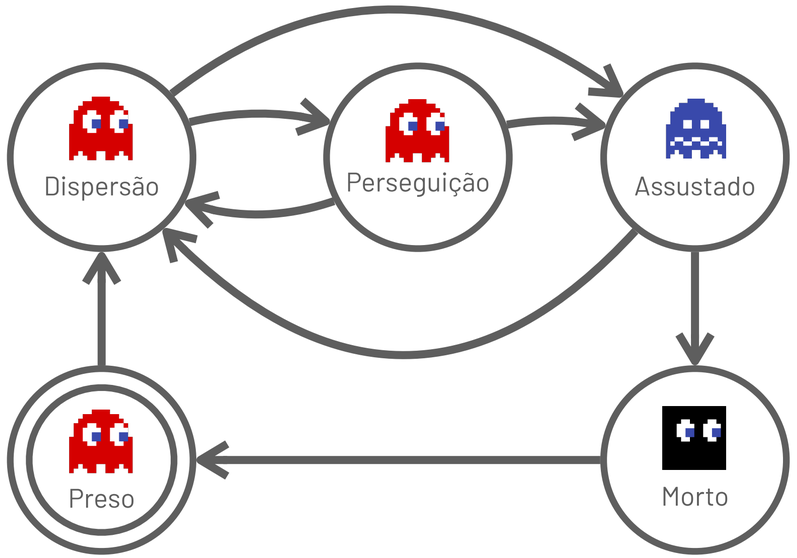



```
enum State {
  STATE_SCATTER,
  STATE_CHASE,
  STATE_FRIGHTED,
  STATE_DEAD,
  STATE_LOCKED
};

switch (state) {
    case STATE_SCATTER:
      break;

    case STATE_CHASE:
      break;

    case STATE_FRIGHTED:
      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(Ghost& g, Input i) {}
  virtual void update(Ghost& g) {}
  virtual void exit(Ghost& g) {}
};

class ScatterState {

public:
  void enter(Input input) {
    state->enter(this, input);
  }

  void update() {
    state->update(this);
  }

  void exit() {
    state->update(this);
  }
}
```

Implementação de FSMs

Interpretadores

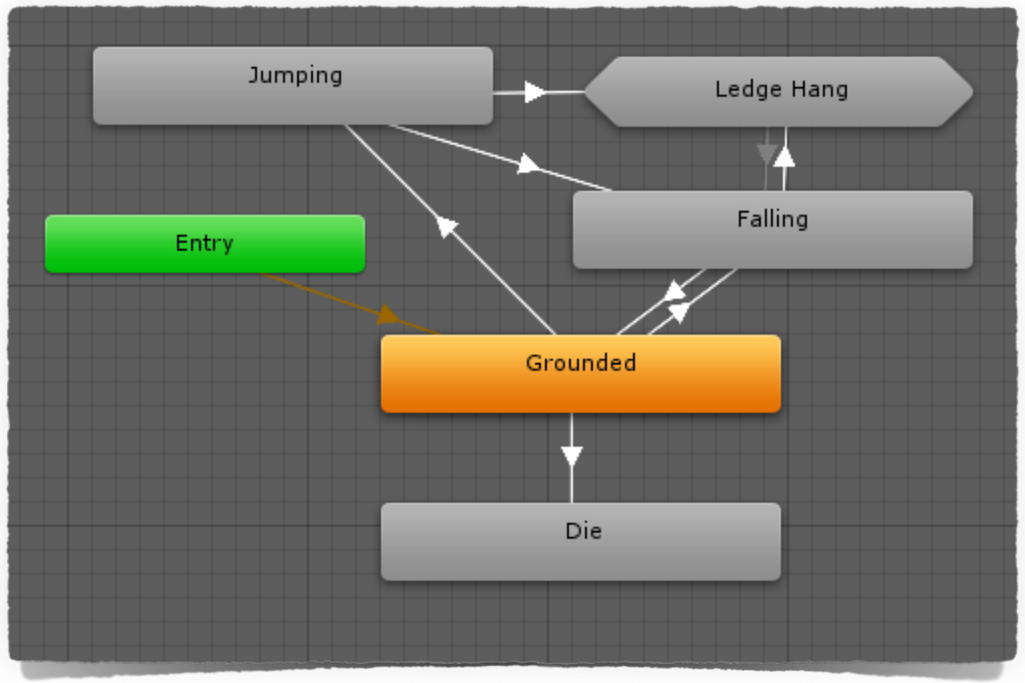


Figura 1: FSM da Unity Engine

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.

Próximas aulas

A9: IA - Parhfinding

Encontrando caminhos mínimos para navegação inteligente.

L9: Pacman - Parte 1

Implementar um componente para máquina de estados.