Inteligência Artificial em Jogos Eletrônicos

Slides por:
Leonardo Tórtoro Pereira (leonardop@usp.br)

Este material é uma criação do
Time de Ensino de Desenvolvimento de Jogos Eletrônicos (TEDJE)
Filiado ao grupo de cultura e extensão Fellowship of the Game (FoG), vinculado ao ICMC - USP

Objetivos

- Introduzir conceito de IA
- Mostrar as principais áreas de utilização de IA em jogos
- Apresentar, conceitualmente e, quando conveniente, com pseudocódigo, os principais algoritmos de cada área
- Apresentar algumas das áreas de pesquisa atuais em IA para jogos

O que é IA?

O que é IA?

- Inteligência exibida por máquinas
- Realiza ações “cognitivas” associadas a humanos ou animais
  - Aprendizado
  - Resolução de problemas

O que é IA?

- 3 motivações do ramo
  - Filosofia
    - Entender natureza do pensamento e da inteligência e construir modelos de *software* para modelar como o pensamento deve funcionar

O que é IA?

- 3 motivações do ramo
  - Psicologia
    - Entender as mecânicas do cérebro humano e processos mentais

O que é IA?

- 3 motivações do ramo
  - Engenharia
    - Construir algoritmos para realizar atividades humanas
  - Este é o foco de interesse nas IAs de jogos

O que é IA?

- Problemas centrais e objetivos da pesquisa de IA
  - Raciocínio
  - Conhecimento
  - Planejamento
  - Aprendizado

O que é IA?

- Problemas centrais e objetivos da pesquisa de IA
  - Processamento de Linguagem Natural
  - Percepção
  - Mover e manipular objetos

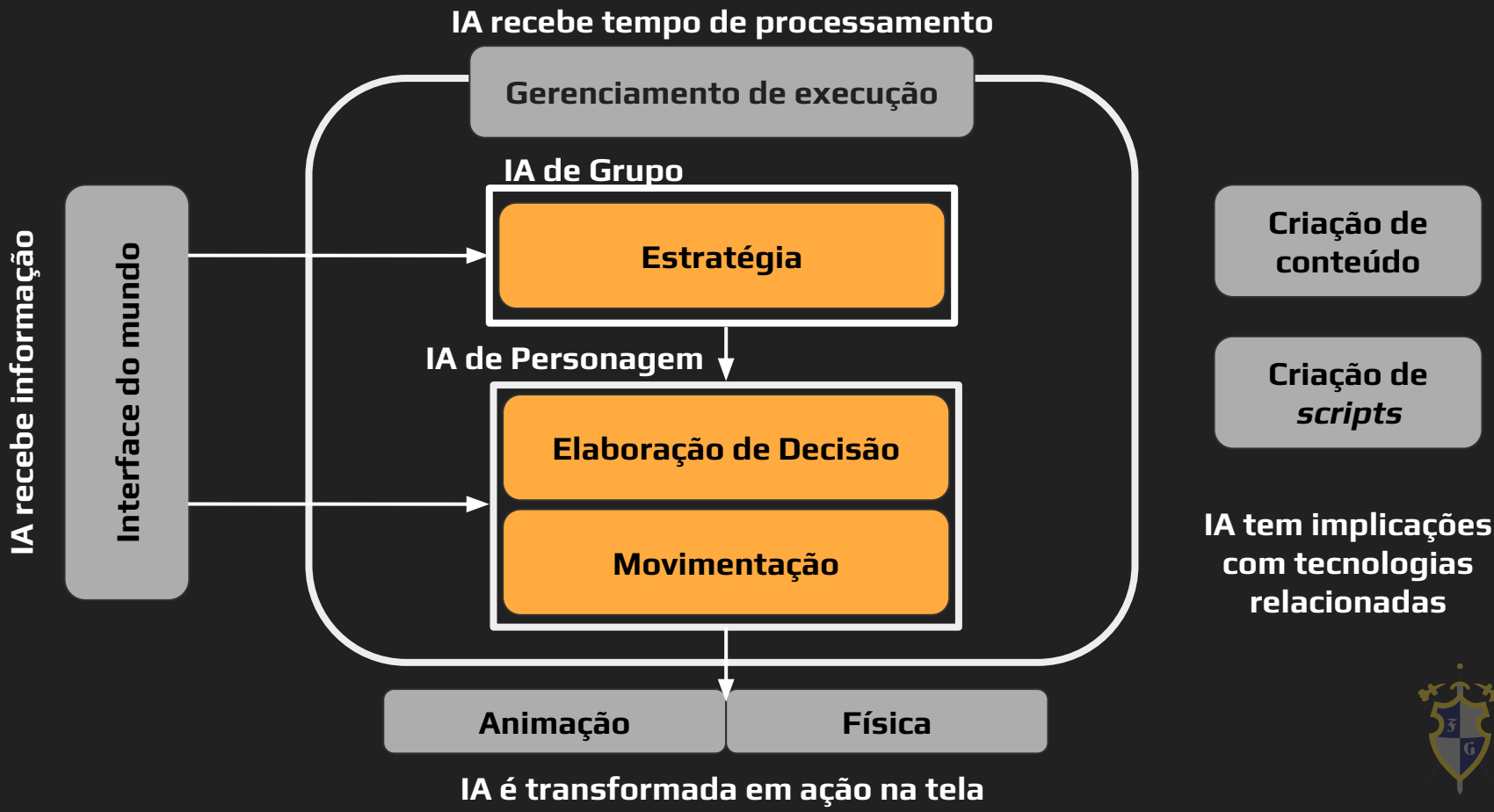
O que é IA?

- Abordagens populares
  - Métodos estatísticos (Cadeias de Markov)
  - Inteligência computacional (AE, AM)
  - IA simbólica (Baseada em lógica)

O que é IA?

- Objetivos
  - Para jogos são 3 os principais
    - Mover personagens
    - Fazer decisões sobre para onde mover
    - Habilidade de pensar taticamente ou estrategicamente

Modelo de IA


IA recebe tempo de processamento
Gerenciamento de execução
IA de Grupo
Estratégia
IA de Personagem
Elaboração de Decisão
Movimentação
IA recebe informação
Interface do mundo
Criação de conteúdo
Criação de *scripts*
IA tem implicações com tecnologias relacionadas
Animação
Física
IA é transformada em ação na tela

Mas por que estudar separadamente para jogos?

???

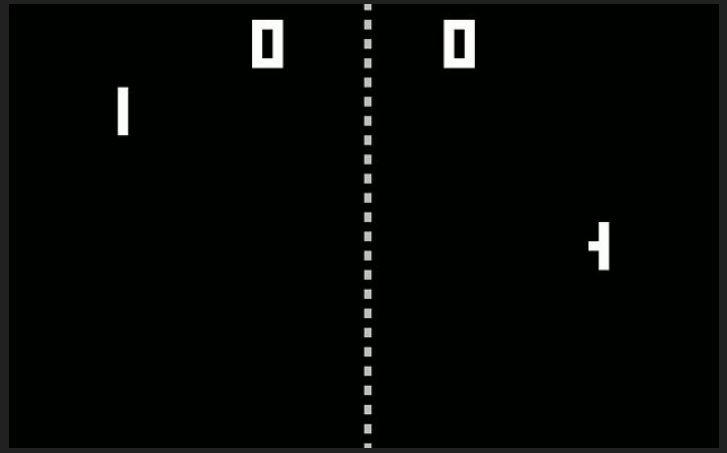
- ➔ Existe foco em velocidade, mas não tanto quanto em sistemas embarcados e aplicações de controle
- ➔ Existe foco em algoritmos inteligentes, mas não com tanto rigor como em servidores de bancos de dados
- ➔ Usa técnicas de muitas fontes diferentes, mas quase sempre muda-as tanto que não são mais reconhecíveis
- ➔ Desenvolvedores fazem modificações de maneiras diferentes, deixando os algoritmos irreconhecíveis de estúdio para estúdio

Breve História da IA em Jogos

Breve História da IA em Jogos

- 1972-1979
  - IAs simples, perseguiam um objeto diretamente
  - Pong, Space Invaders

Pong



Breve História da IA em Jogos

- 1979-1996
  - Máquina de estados
  - Pac-Man (1979)
    - Para cada estado o fantasma escolhia uma rota semi-aleatória em cada junção

Pac-Man

Breve História da IA em Jogos

- 1979-1996
  - Rogue (1980)
    - Dungeons gerados procedimentalmente

Rogue

Breve História da IA em Jogos

- 1997-Atualidade
  - Personagens com simulação de sentidos
  - Inimigos sabiam quando aliados morriam
  - Goldeneye 007 (1997), Thief: The Dark Project (1998) e Metal Gear Solid (1998)

Goldeneye 007

Breve História da IA em Jogos

- 1997-Atualidade
  - Algoritmos de buscas de rota (*pathfinding*) ganham destaque com *Warcraft* (1994), mas já haviam sido aplicado várias vezes anteriormente

Warcraft: Orcs & Humans

Breve História da IA em Jogos

- 1997-Atualidade
  - Pesquisadores de IA estavam pesquisando modelos emocionais de soldados em simulação de campo de batalha militar, quando encontraram Warhammer: Dark Omen (1998) fazendo a mesma coisa
  - Também foi a primeira vez que foi visto movimentação de formação robusta

Warhammer: Dark Omen

Breve História da IA em Jogos

- 1997-Atualidade
  - IA começou a ser um importante ponto de venda do jogo
  - *Creatures* (1997) possuía um cérebro baseado em rede neural para cada criatura (que normalmente mostrava-se bem burra em ação)

Creatures


Creatures - <Un Named>
File View Camera Tools Norns Help
For Help, press F1
BIRTH
Life Force 73%
Score 1
start
Creatures

Breve História da IA em Jogos

- 1997-Atualidade
  - Geração procedimental de conteúdo
  - IAs que dirigem o ritmo do jogo e o conteúdo a cada momento
  - *The Director* - Left 4 Dead (2008)

Breve História da IA em Jogos

- 1997-Atualidade
  - Competições de *bots* de IA e de geração de conteúdo (*Starcraft, Mario, Angry Birds*)
  - Diversos congressos focados na área
  - Pesquisas amplamente divulgadas e de “fácil” acesso
  - Até *youtubers* estão ficando famosos com experimentos
  - Veremos exemplos no final da aula!

IA em Jogos

IA em Jogos

- Quanto mais complexa a IA
  - **Não** necessariamente melhor o jogo
- IAs simples podem deixar o jogo muito bom
- Por exemplo: *Pac-Man*

IA em Jogos

- Pac-Man
  - IA tem dois estados:
    - Jogador colecionando “bolinhas”
    - Jogador comeu o *power-up*

IA em Jogos

- No estado normal
  - Fantasmas movem-se em linha reta até alcançar junção
  - Nela, eles escolhem semi-aleatoriamente a próxima rota
  - Cada fantasma escolhe pegar ou a rota que é na direção do jogador (*offset* da posição do jogador)

IA em Jogos

- No estado normal
  - Ou rota aleatória
  - Cada fantasma tem uma probabilidade de escolher cada rota
  - Mais simples que isso os fantasmas seriam previsíveis ou totalmente aleatórios
  - Probabilidade dá personalidade às IAs

IA em Jogos

- Janela de percepção
  - O que sua IA pensa deve ser transmitida imediatamente ao jogador
    - Se ouve um som, tem que ser claro o que fará
      - Virar para a fonte
      - Acender a luz
      - Dar um alarme

IA em Jogos

- Janela de percepção
  - A intenção de ação deve ser mostrada ao jogador
  - Ou ele achará a IA mal implementada
  - Por exemplo:
    - Se a chave de luz estiver longe e o jogador matar o inimigo rapidamente, o jogador poderá pensar que o inimigo não reagiu ao som e morreu por ser burro

IA em Jogos

- Mudanças de comportamento
- São facilmente notadas
  - Assim como a ausência delas, quando deveriam existir
- É comum existir uma com a proximidade do jogador

Tipos de IA em jogos

Tipos de IA em Jogos

- Para desenvolvedores de jogos, existem 3 tipos de IAs
  - *Hacks*: soluções *ad hoc*(gambiarra) e efeitos legais
  - Heurísticas: Regras de ouro (achismo)
  - Algoritmos: Cientificamente comprovados

Tipos de IA em Jogos

- *Hacks*
  - “Se parece com um peixe e cheira como um, provavelmente é um peixe”
  - Pense no *design* do personagem e faça o possível para ele parecer com o que foi planejado
  - Muitas vezes, é preciso apenas uma animação específica, ou até um ato aleatório
  - Você pode perceber a diferença, o jogador (talvez) não

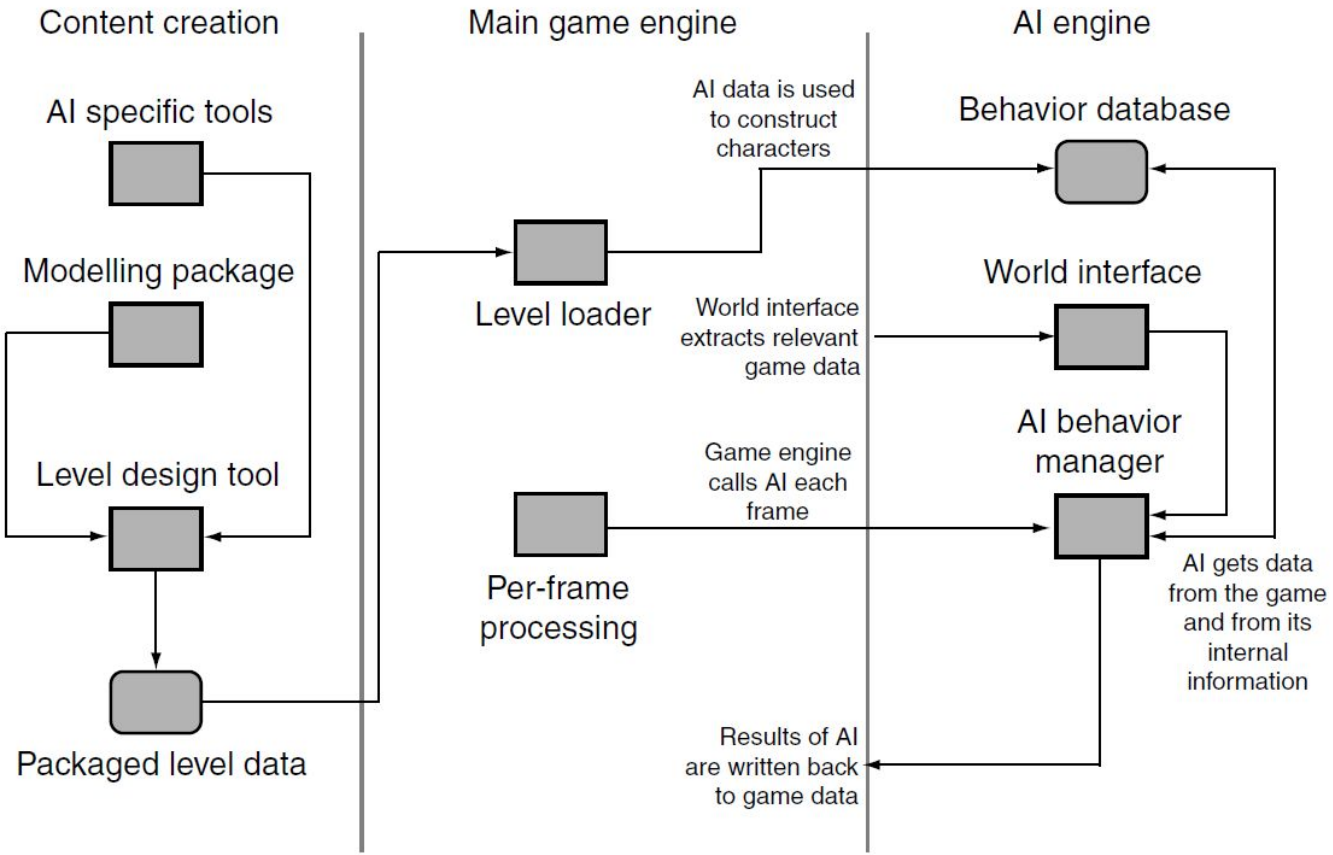
Tipos de IA em Jogos

- ➔ Heurísticas
  - Solução aproximada que pode funcionar na maioria das situações
    - Talvez não em todas (¯_(ツ)_/¯)
  - Humanos usam o tempo todo
  - Movimento dos fantasmas em Pac-Man
  - Dar um número para qualificar peças em jogos de estratégia

Engine de IA

Engine de IA

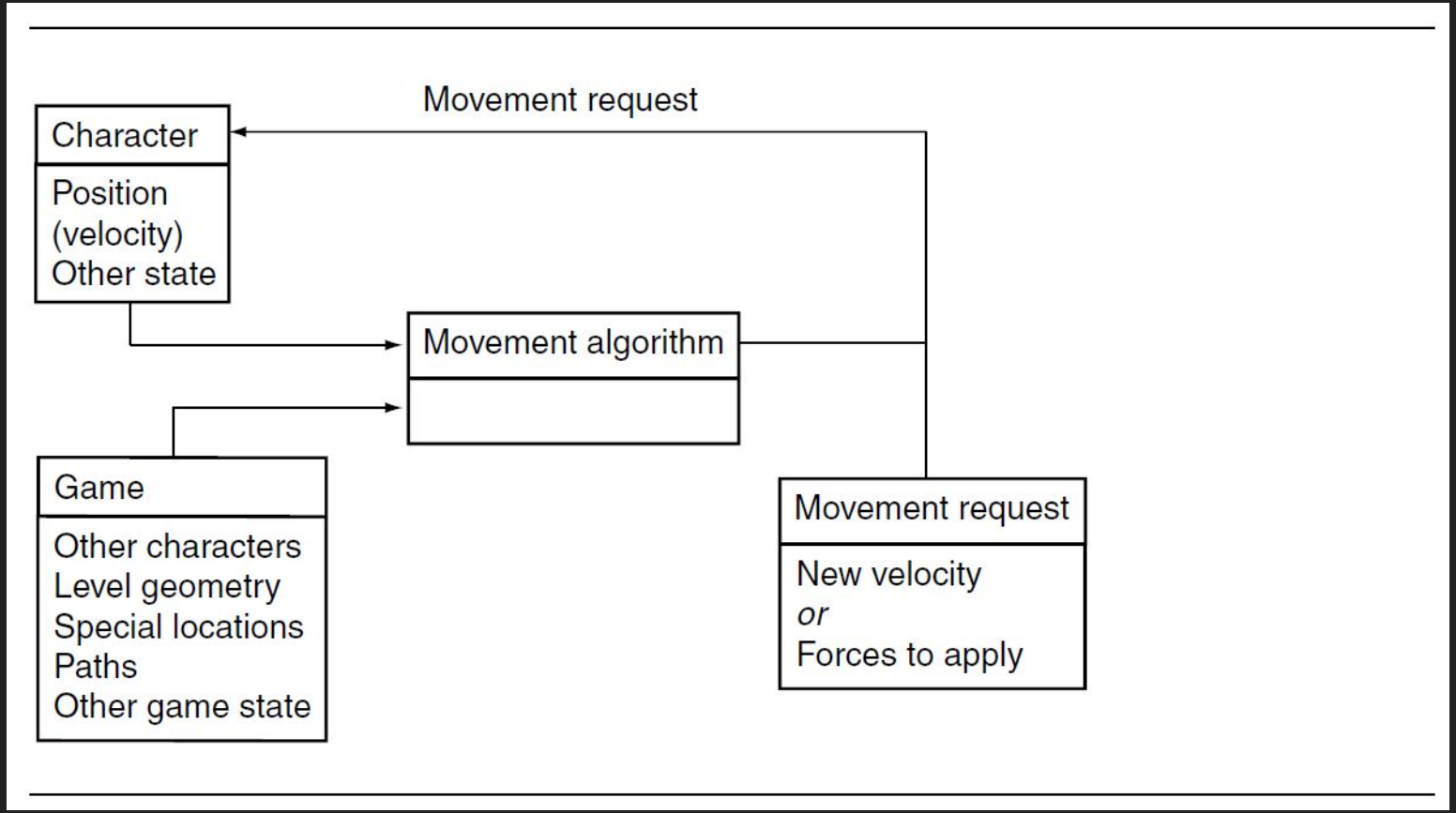
- ➔ Com a crescente complexidade dos jogos, reutilizar IAs torna-se essencial
- ➔ A seguir temos uma proposta de modularização de IAs para facilitar a reutilização



Fonte: Artificial Intelligence for Games

Movimento

Estrutura de algoritmo de movimento



Fonte: Artificial Intelligence for Games

Movimento

- Movimentação pode ser 2D ou 3D
- Existem diferentes formas de calcular a movimentação linear e angular do personagem
- Uso de comportamentos de *steering*
  - Dinâmicos, ou seja, considerando a movimentação atual do personagem

```
struct Kinematic
  position    # a 2 or 3D vector
  orientation # a single floating point value
  velocity    # another 2 or 3D vector
  rotation    # a single floating point value
```

```
struct SteeringOutput:
  linear    # a 2 or 3D vector
  angular   # a single floating point value
```

Fonte: Artificial Intelligence for Games

Movimento

➔ Veremos a seguir o *update* de posição (2D), usando cinemática, com a integração de *Newton-Euler-1*, muito usada pela praticidade

```
struct Kinematic:

  ... Member data as before ...

  def update (steering, time):

    # Update the position and orientation
    position += velocity * time
    orientation += rotation * time

    # and the velocity and rotation
    velocity += steering.linear * time
    orientation += steering.angular * time
```

Fonte: Artificial Intelligence for Games

Movimento

- Existem vários comportamentos de movimentação diferentes
- Busca, fuga, alinhamento, chegada, vagar, etc.
- Não serão apresentados nesta aula
- Mas estão no material de referência
- Falemos brevemente sobre movimentação de projéteis

Movimento

- Devido à característica discreta dos jogos, o simples cálculo do arremesso de um projétil é uma tarefa que envolve diversos cálculos e simplificações físicas
- Vamos apresentar um exemplo de código resultante destas simplificações, no qual queremos encontrar o vetor correto de direção do disparo
- O passo a passo pode ser encontrado na referência

Movimento

- Equação para determinar o vetor de disparo

$$\vec{u} = \frac{2\vec{\Delta} - \vec{g}t_i^2}{2s_m t_i}.$$

- Onde Δ é o vetor do ponto inicial ao final (E-S), g é o vetor da gravidade, s_m é a velocidade do projétil e t_i é o tempo de chegada do projétil ao alvo

```
def calculateFiringSolution(start, end, muzzle_v, gravity):

  # Calculate the vector from the target back to the start
  delta = start - end

  # Calculate the real-valued a,b,c coefficients of a conventional
  # quadratic equation
  a = gravity * gravity
  b = -4 * (gravity * delta + muzzle_v*muzzle_v)
  c = 4 * delta * delta

  # Check for no real solutions
  if 4*a*c > b*b: return None

  # Find the candidate times
  time0 = sqrt((-b + sqrt(b*b-4*a*c)) / (2*a))
  time1 = sqrt((-b - sqrt(b*b-4*a*c)) / (2*a))

  # Find the time to target
  if times0 < 0:
```

Fonte: Artificial Intelligence for Games

```
    if times1 < 0:
      # We have no valid times
      return None
    else:
      ttt = times1
  else:
    if times1 < 0:
      ttt = times0
    else:
      ttt = min(times0, times1)

  # Return the firing vector
  return (2 * delta - gravity * ttt*ttt) / (2 * muzzle_v * ttt)
```

Fonte: Artificial Intelligence for Games

Movimento

- Algoritmos para cálculo de pulos
- Algoritmos para movimentação coordenada (em formação)
- Algoritmos para movimentação tática
- Algoritmos para movimentação de veículos motorizados
- Algoritmos para movimentação 3D
  - Eixos de rotação

Pathfinding

Pathfinding

- ➔ Planejamento de movimento entre pontos distantes
- ➔ “Roteador”
- ➔ Rota deve ser o mais sensata e curta possível

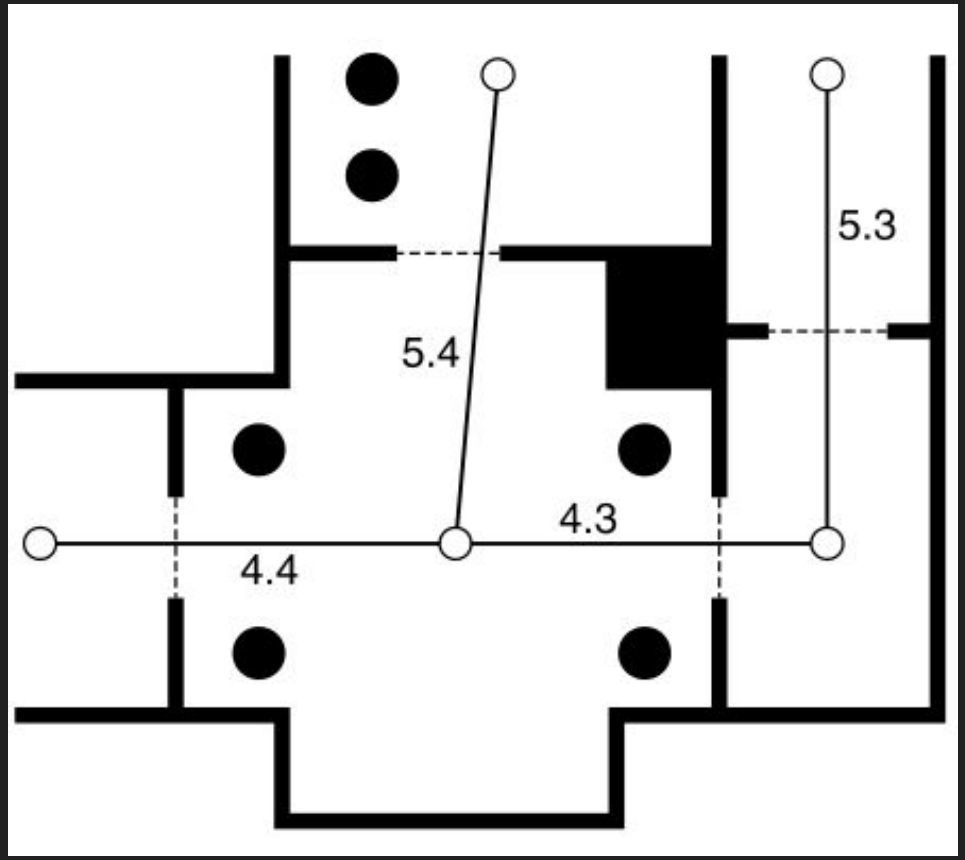
Pathfinding

- A maioria dos jogos usa o A* (que veremos com detalhes)
  - Eficiente e fácil de implementar
  - Não trabalha diretamente com os dados de nível do jogo
  - Requer que o nível seja representado como grafo direcionado com pesos não negativos
  - “Primo” do algoritmo de Dijkstra

Pathfinding

- ➔ Tanto Dijkstra quanto A* (e outros grafos) usam uma simplificação do nível
  - ◆ Representada em grafo
  - ◆ Se bem simplificada
    - ● O plano retornado pelo algoritmo será útil quando traduzido em termos de jogos
  - ◆ Se simplificação descartar informações relevantes
    - ● Plano pode ser ruim

Grafo ponderado sobreposto à geometria de um nível



Fonte: Artificial Intelligence for Games

*Pathfinding - A**

*Pathfinding - A**

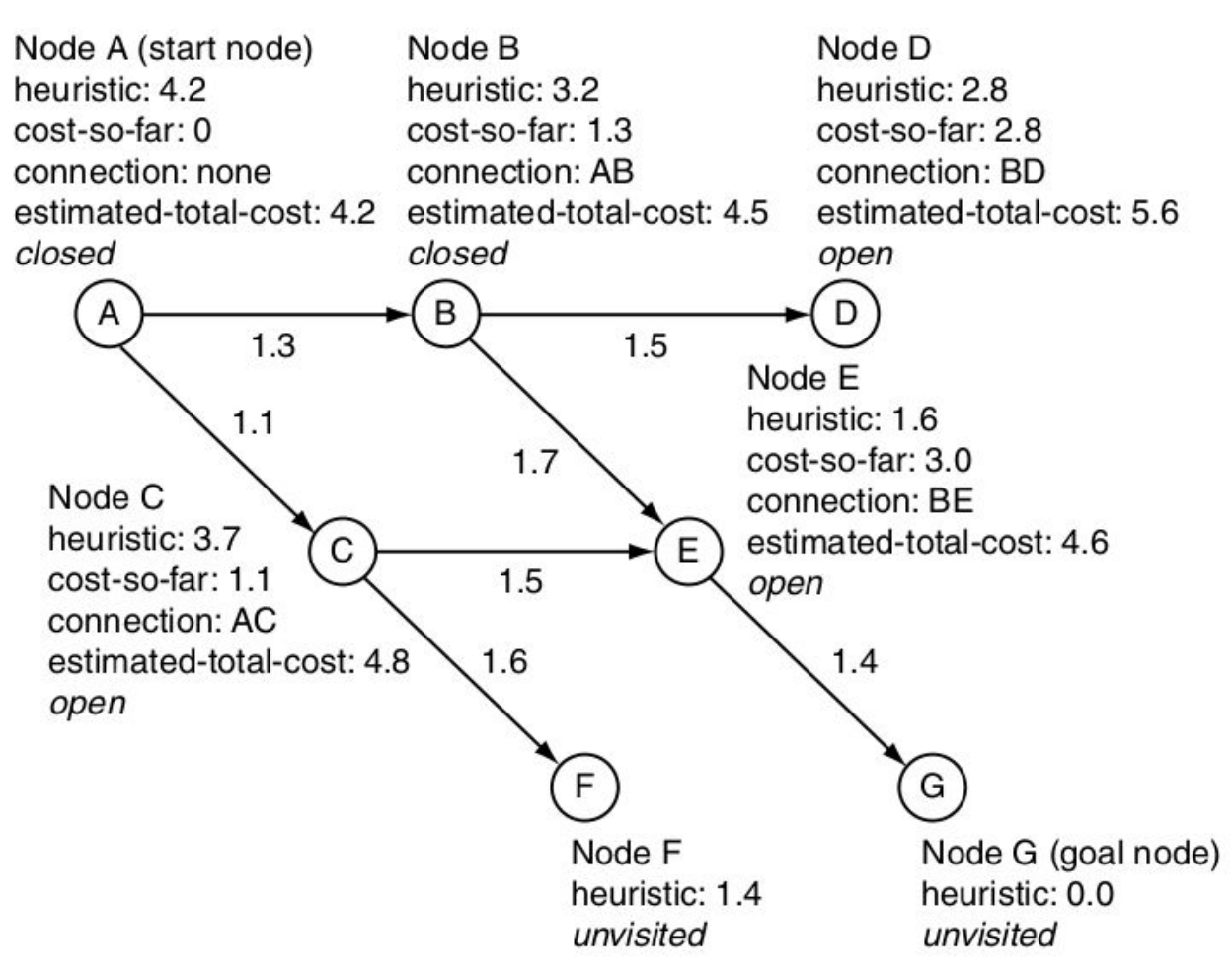
- Simples de implementar
- Muito eficiente
- Grande espaço para otimização
- Encontra caminho ponto-a-ponto
  - E não caminho mínimo
- Ao invés de considerar o nó visitado com o valor mais baixo até o momento, considera aquele que provavelmente levará ao melhor caminho (heurística)

*Pathfinding - A**

- Depende muito da heurística escolhida
- Trabalha por iterações
  - A cada iteração considera 1 nó do grafo e segue suas conexões de saída
  - O nó (chamado de nó atual) é escolhido através de um algoritmo de seleção similar ao de Dijkstra, mas com a diferença da heurística

*Pathfinding - A**

- Em cada iteração, para cada conexão de saída do nó é encontrado o nó final, e o valor total do custo do caminho até agora é armazenado, juntamente com a conexão com a qual o algoritmo chegou até ele.
- Além disso, o custo total estimado para o caminho do nó inicial até este nó, e deste nó até o objetivo.
  - Soma do custo do caminho até agora e o quão longe está do objetivo (heurística)



Fonte: Artificial Intelligence for Games

*Pathfinding - A**

- ➔ O algoritmo mantém uma lista de nós abertos já visitados mas não processados, e nós fechados (já processados)
- ➔ Nós são movidos para a lista de abertos conforme são encontrados no fim das conexões
- ➔ Nós são movidos para a lista de fechados conforme são processados em sua iteração

*Pathfinding - A**

- O nó da lista de abertos com o menor custo total estimado é selecionado a cada iteração
  - Quase sempre é diferente do nó com o menor custo até o momento
- Permite ao algoritmo examinar nós que são mais promissores primeiro

*Pathfinding - A**

- Se a heurística for boa, os nós mais pertos do objetivo são considerados primeiro, estreitando a busca para a área mais promissora
- Ao chegar em um nó aberto ou fechado, é necessário revisar seus valores
- Calcula-se o custo até agora normalmente
  - Se for mais baixo que o valor existente no nó, é necessário atualizá-lo (não usar custo estimado)

*Pathfinding - A**

- Diferente de Dijkstra, o A* pode achar rotas melhores para nós que estão na lista de fechados.
- Se uma estimativa prévia for muito otimista, o nó pode ter sido processado pensando-se que era a melhor escolha quando, de fato, não o era
  - Isso pode causar um problema em cadeia

*Pathfinding - A**

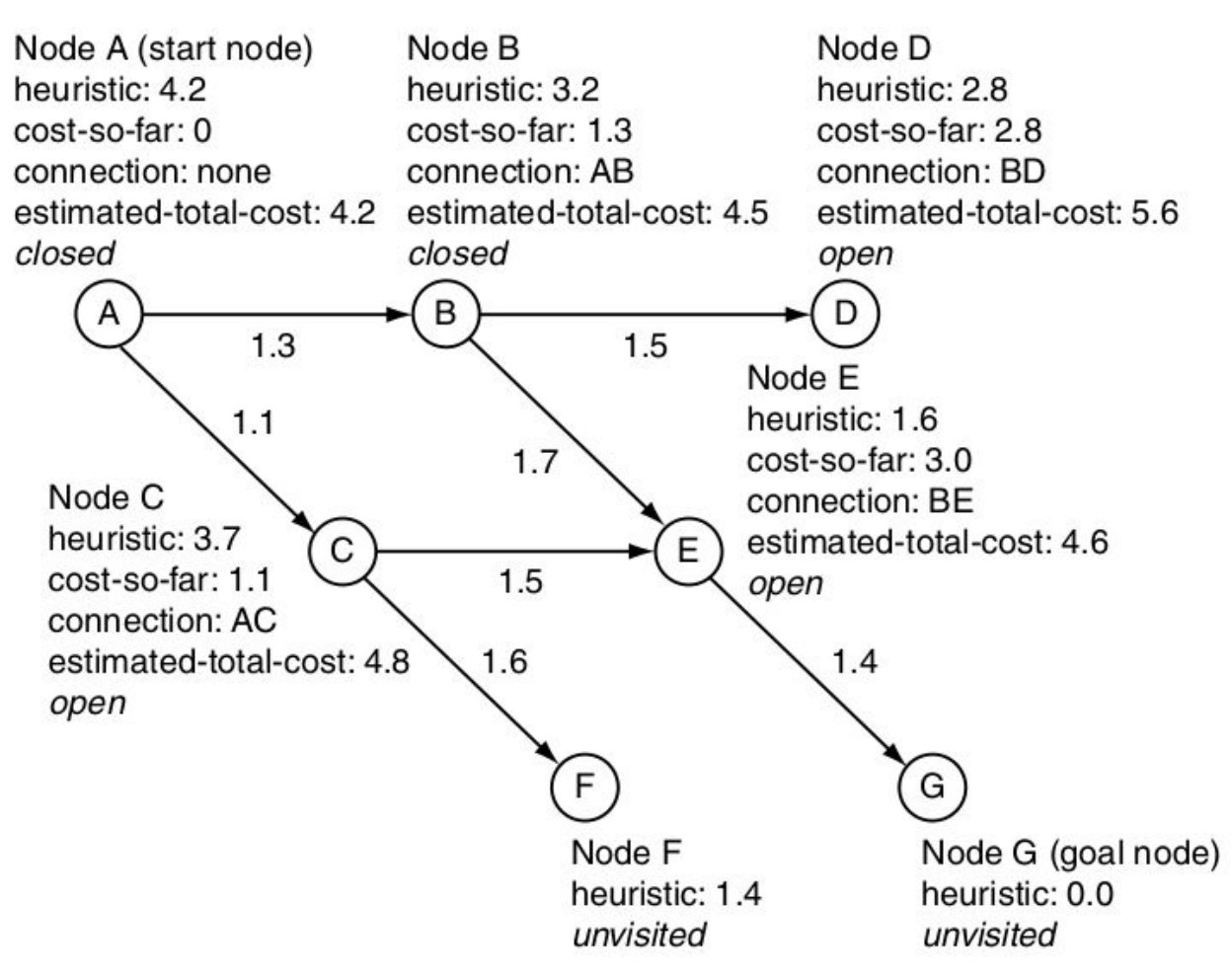
- Se um nó teve uma estimativa errada e colocado na lista de fechados, todas as suas conexões foram consideradas
- Pode ser que um conjunto de nós tiveram seus custos até agora baseado em um custo até agora de um nó errado
- Portanto, atualizar o valor do nó errado não é o suficiente
  - Todas suas conexões devem ser checadas novamente para propagar o valor

*Pathfinding - A**

- ➔ No caso de revisar um nó na lista de abertos, isso não é necessário, uma vez que sabemos que conexões de um nó na lista não foram processadas ainda
- ➔ Existe um jeito simples de forçar o cálculo e propagação do valor
- ➔ O nó pode ser removido da lista de fechados e colocado de volta na de abertos

*Pathfinding - A**

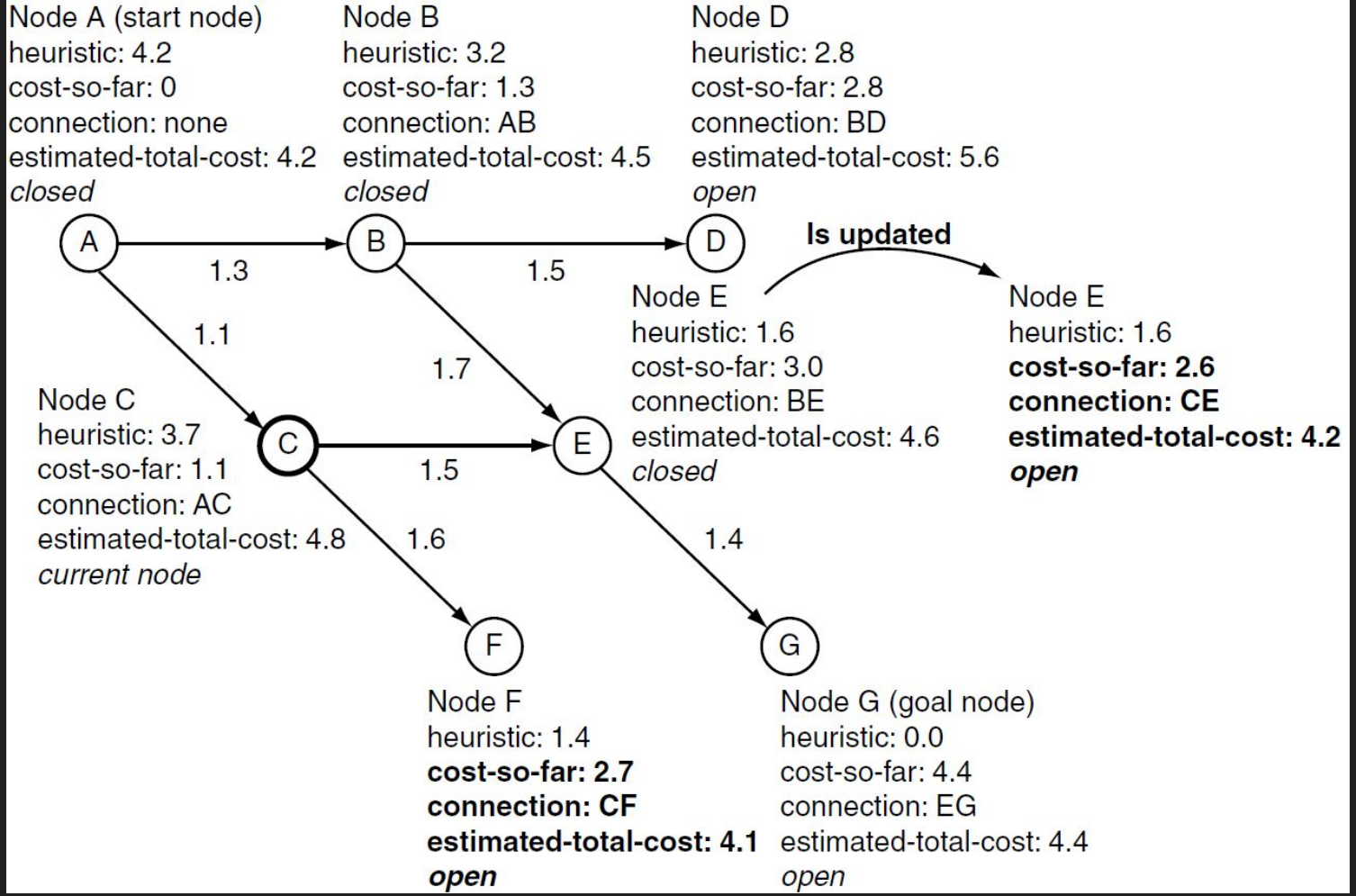
- ➔ Então esperará até ser fechado e ter suas conexões reconsideradas
- ➔ Quaisquer nós que dependam de seu valor serão eventualmente processados mais uma vez
- ➔ Lembra do grafo anterior?



Fonte: Artificial Intelligence for Games

*Pathfinding - A**

- ➔ Este aqui é ele, 2 iterações depois
- ➔ Mostra a atualização de um nó fechado no grafo
- ➔ A nova rota E, via C, é mais rápida



Fonte: Artificial Intelligence for Games

A*

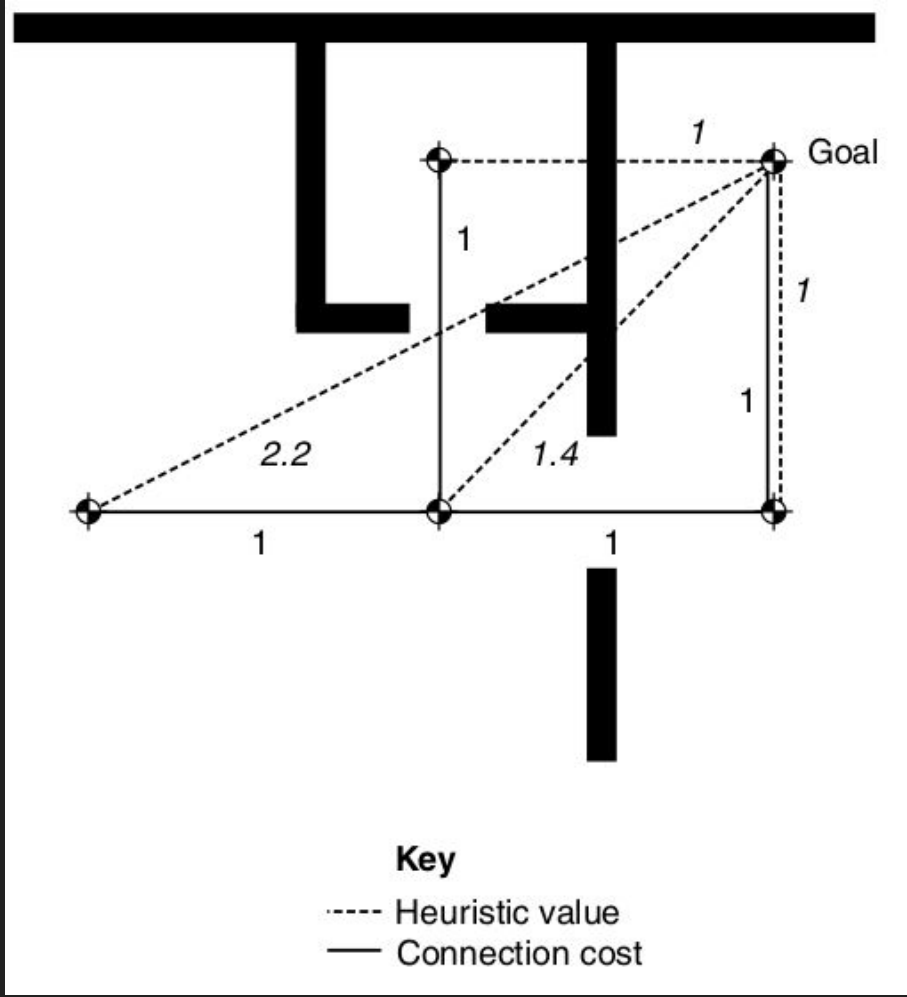
http://www.policyalmanac.org/games/aStarTutorial.htm

https://qiao.github.io/PathFinding.js/visual/

Pathfinding - A*

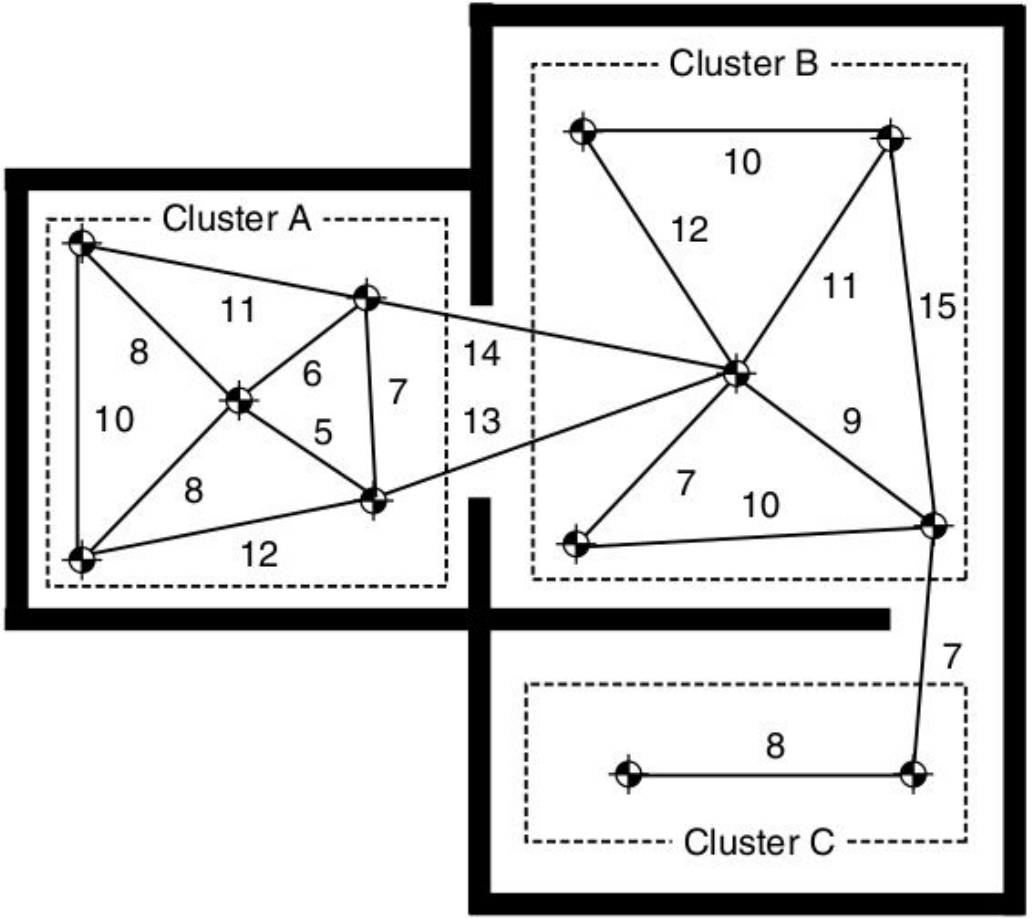
➔ Um pouco sobre heurísticas

Euclidiana



Fonte: Artificial Intelligence for Games

Cluster



	A	B	C
A	x	13	29
B	13	x	7
C	29	7	x

Lookup table

Fonte: Artificial Intelligence for Games

Representação de Mundo

Representação de Mundo

- ➔ É preciso mudar a representação do mundo para os nós e conexões do grafo e a função de custo que dá seus valores
- ➔ Os diferentes modos de dividir o nível em regiões conectadas são chamadas de esquemas de divisão

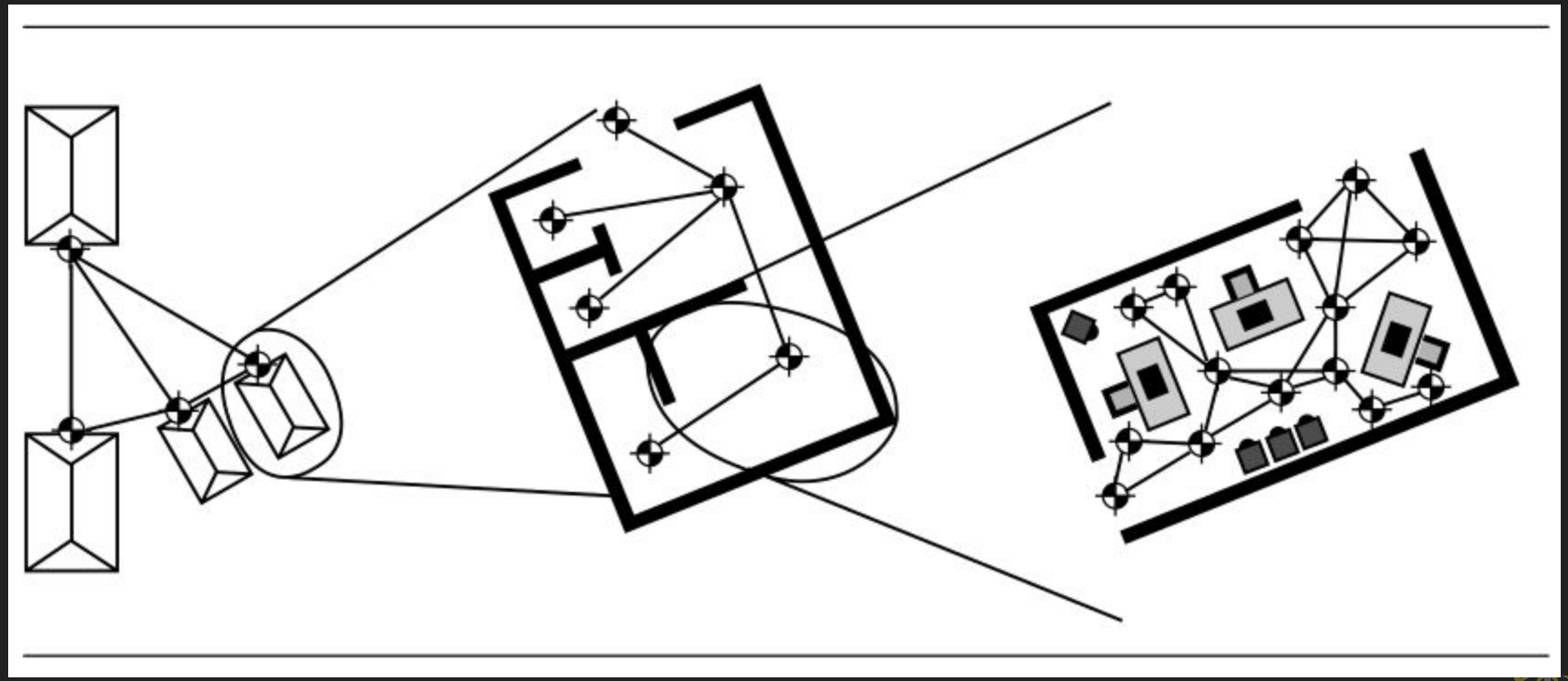
Representação de Mundo

- Cada esquema de divisão tem 3 propriedades importantes
  - Quantização/Localização
  - Geração
  - Validade
- Existem diversos tipos de esquema com suas vantagens e desvantagens
- Não serão abordados nesta aula

Grafo de *Pathfinding* Hierárquico

Grafo de Pathfinding Hierárquico

- Formação de *clusters* de lugares
- Processo similar ao de colisão (*Quad-trees* e *Octa-trees*)
-



Fonte: Artificial Intelligence for Games

Pathfinding - mais ideias

Pathfinding - mais ideias

- ➔ *Pathfinders* podem ser dinâmicos (consideram mudanças no ambiente)
- ➔ E até contínuos (de difícil implementação)

Tomada de Decisão

Tomada de Decisão

- Habilidade de decidir o que fazer
- Simples
  - Máquinas de estado e árvores de decisão
  - Baseado em regras são mais raros
- Complexos
  - Lógica *fuzzy* e redes neurais

Tomada de Decisão

- Personagem possui informações (externas e internas)
  - São usadas para gerar uma ação (interna ou externa)
- Informações externas (ambiente do jogo)
  - Posição de outros personagens
  - Disposição do nível
  - Se algo foi lançado
  - Direção da fonte de um som ou luz
  - Etc

Tomada de Decisão

- Informações internas
  - Vida do personagem
  - Objetivos do personagem
  - Qual a atividade atual
  - Etc.

Tomada de Decisão

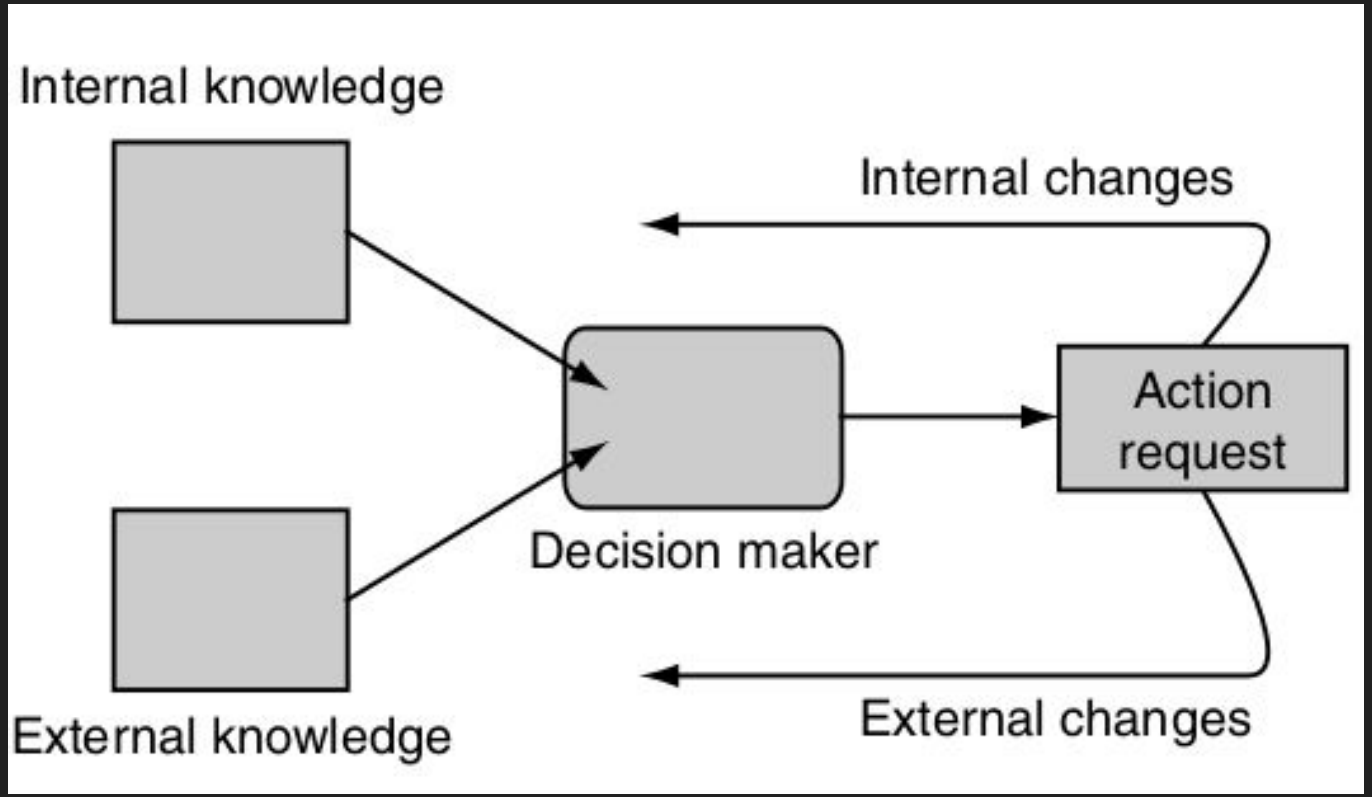
- Ações externas
  - Jogar algo
  - Atirar
  - Ir para uma sala

Tomada de Decisão

- Ações internas
  - Mudar opinião do personagem sobre o jogador
  - Mudança de estado emocional
  - Mudança de objetivo

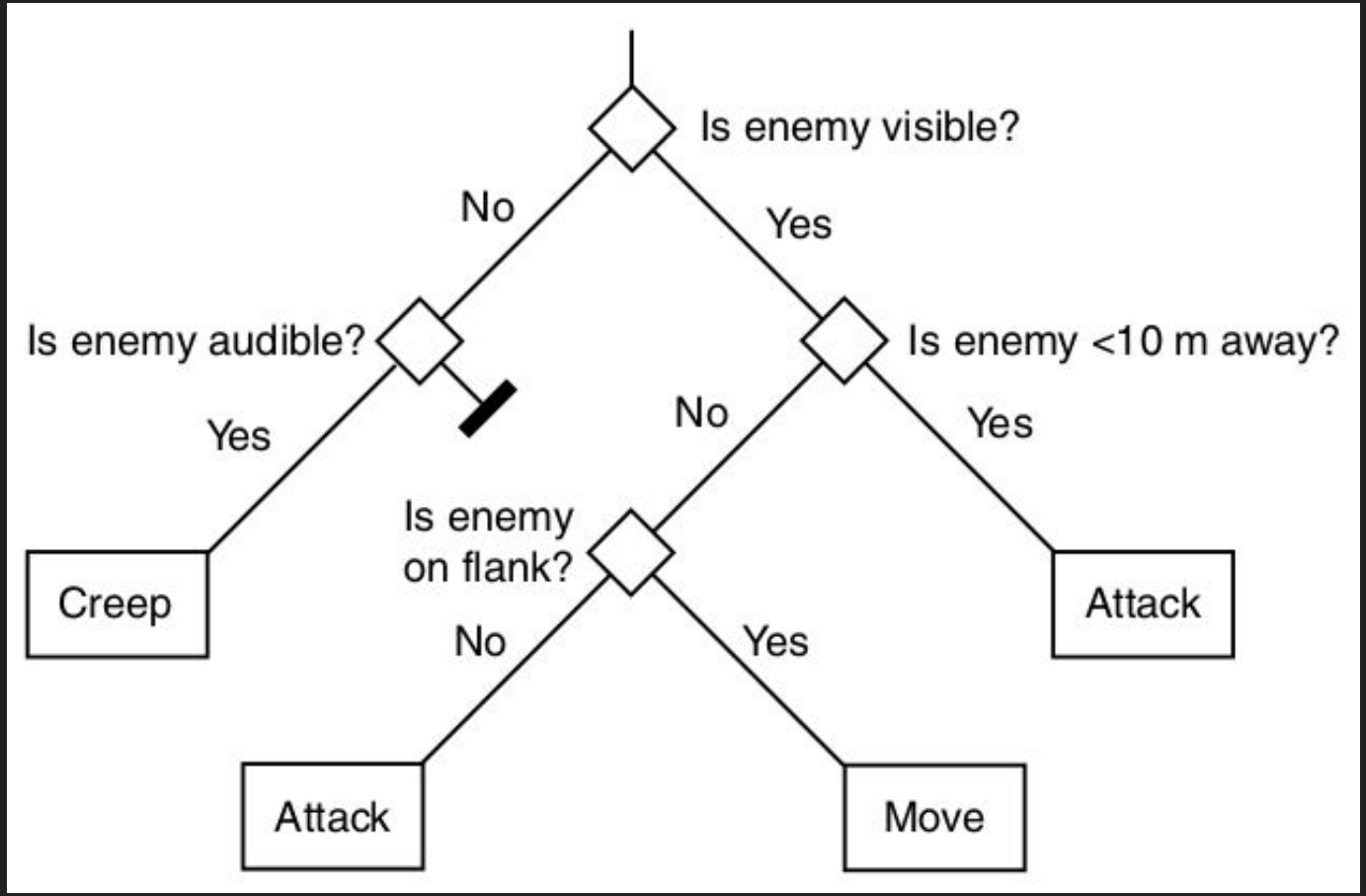
Tomada de Decisão

- Informações internas
  - Vida do personagem
  - Objetivos do personagem
  - Qual a atividade atual
  - Etc.



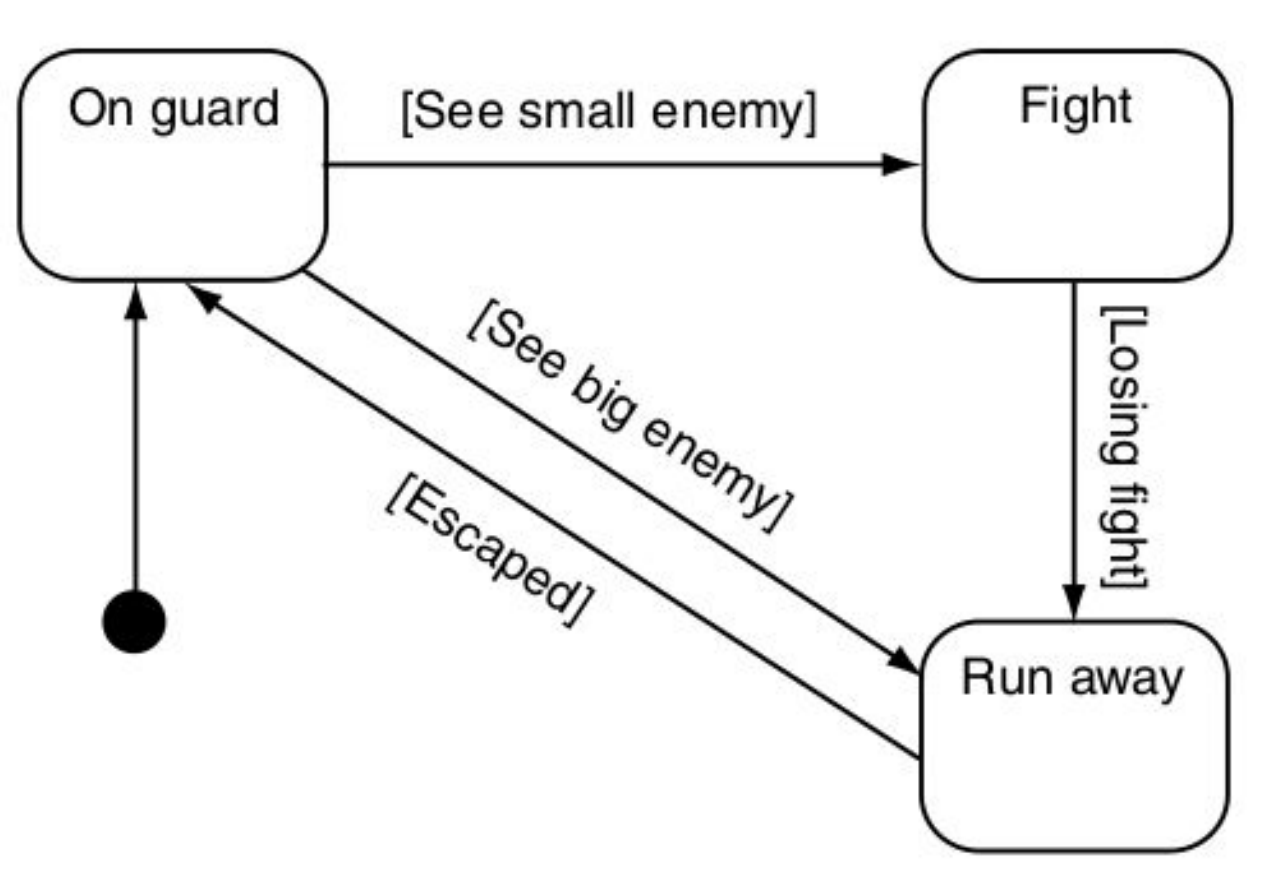
Fonte: Artificial Intelligence for Games

Árvores de Decisão

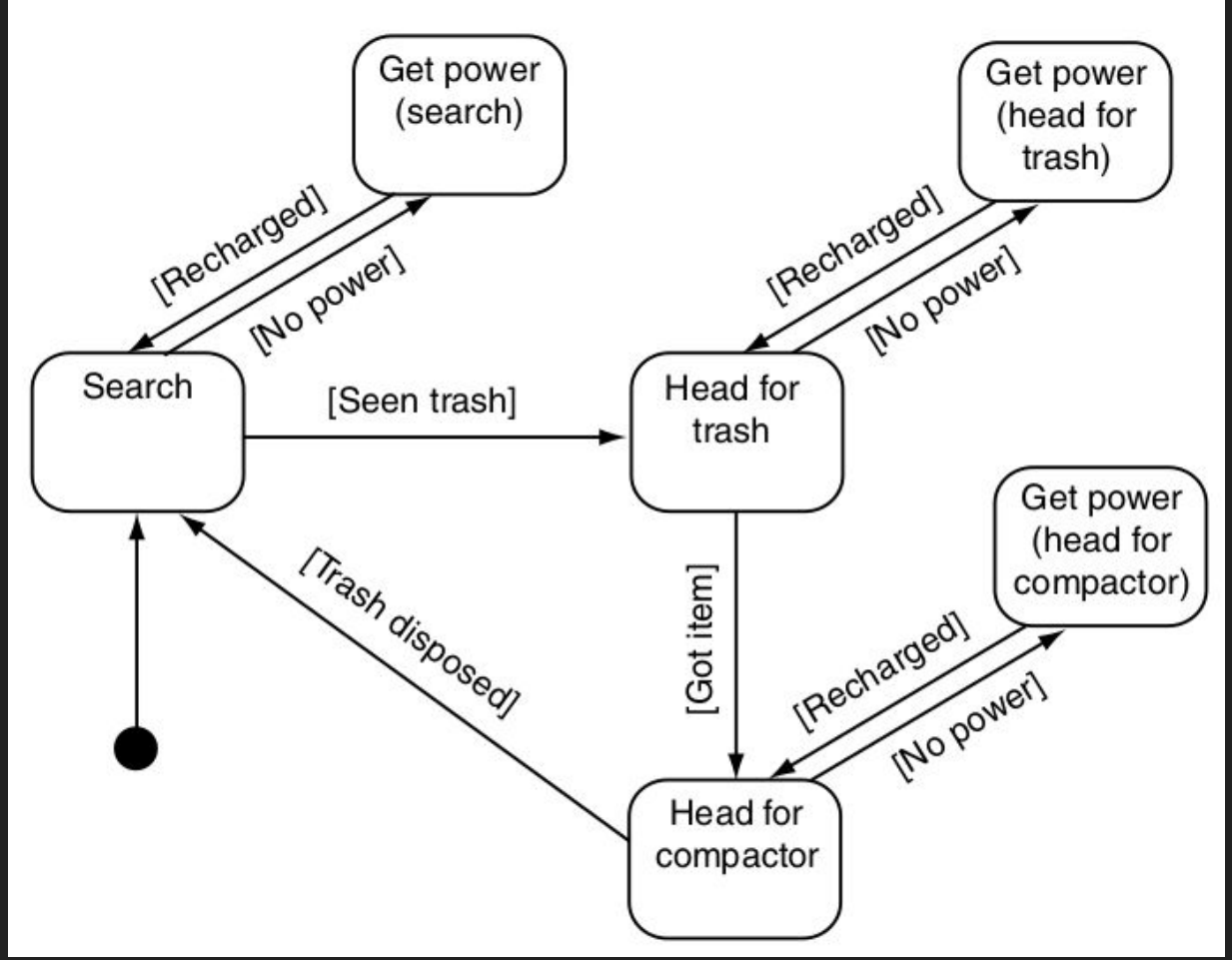


Fonte: Artificial Intelligence for Games

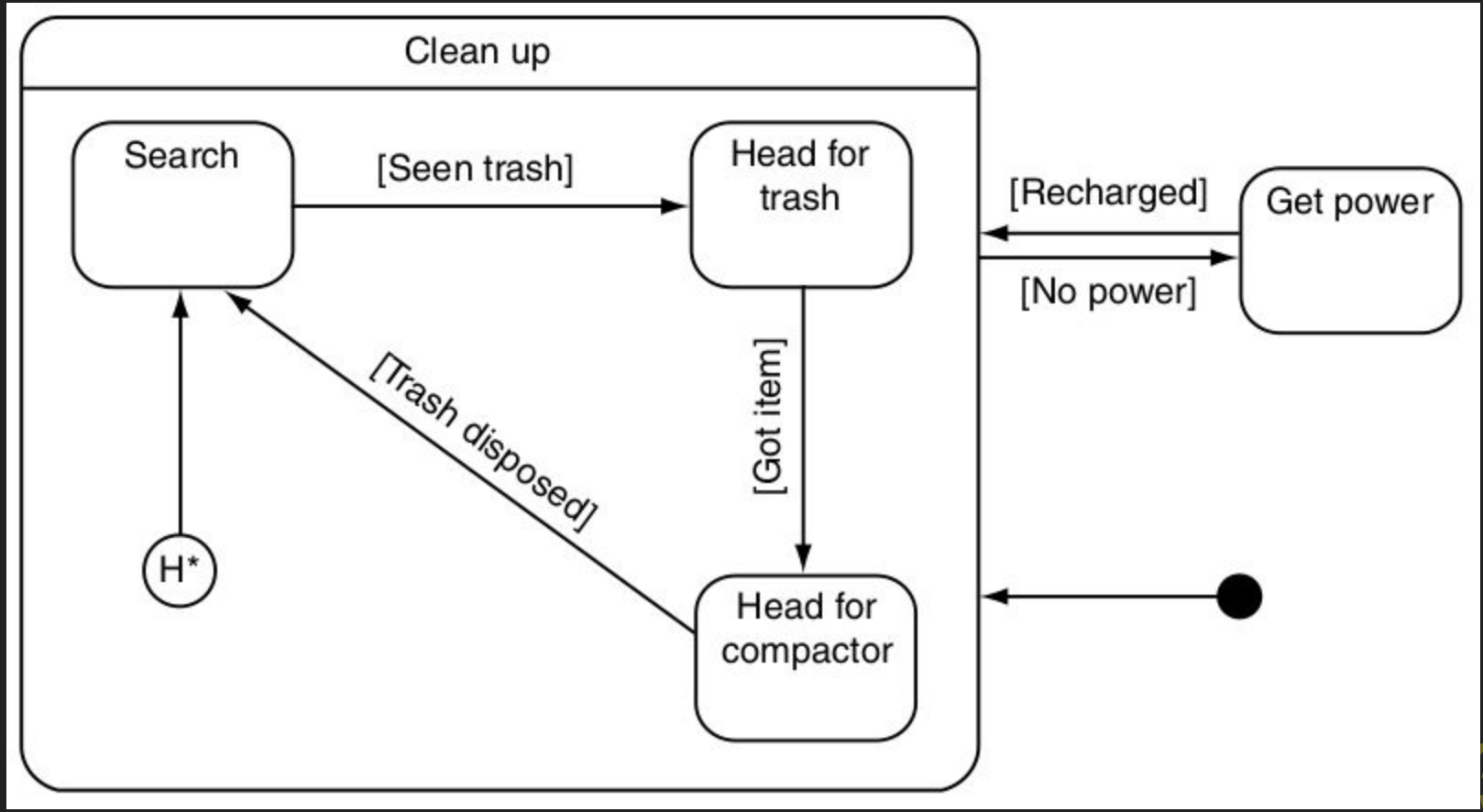
Máquinas de Estado



Fonte: Artificial Intelligence for Games



Fonte: Artificial Intelligence for Games



Fonte: Artificial Intelligence for Games

Lógica Fuzzy

Lógica *Fuzzy*

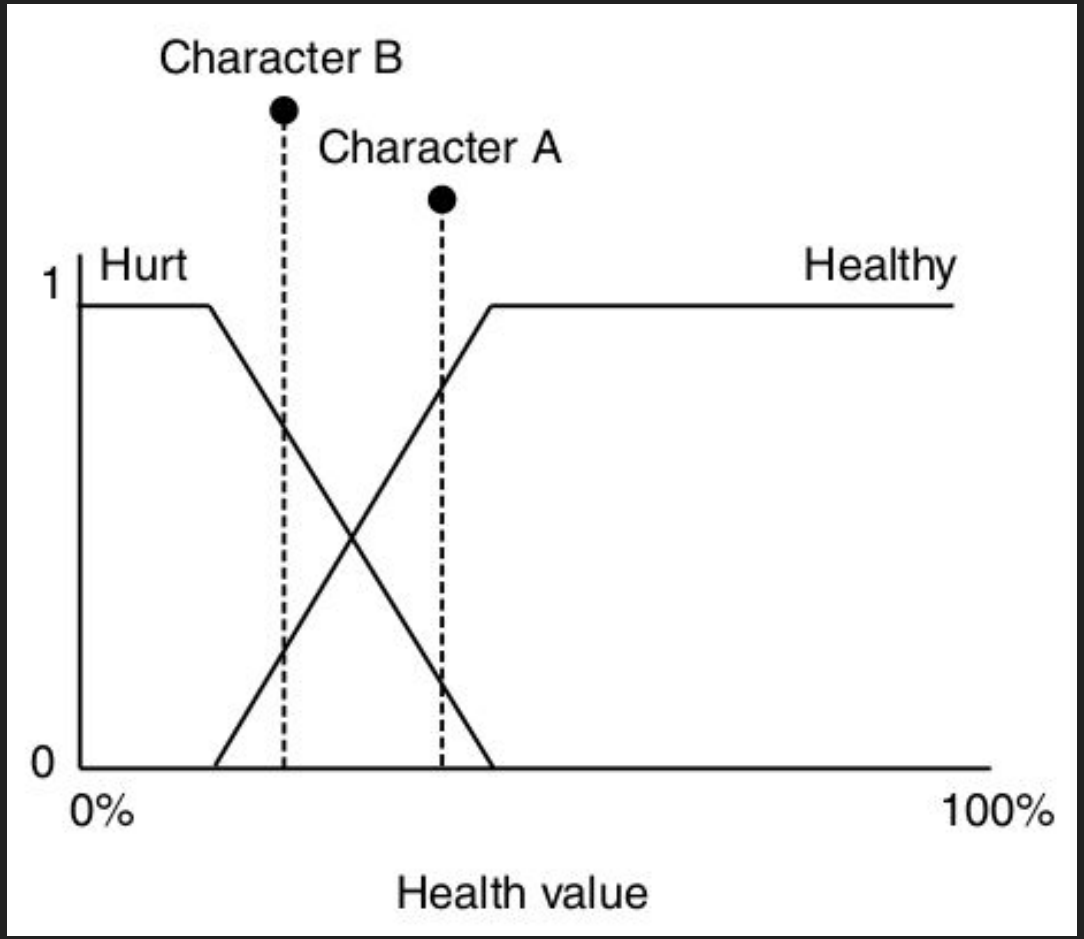
- Conjunto de técnicas matemáticas que lida com decisões não binárias
  - “Área cinza”
- Pode fazer uma transição em vários graus entre cuidadoso e confiante, por exemplo
  - Várias velocidades e níveis de atenção diferentes

Lógica *Fuzzy*

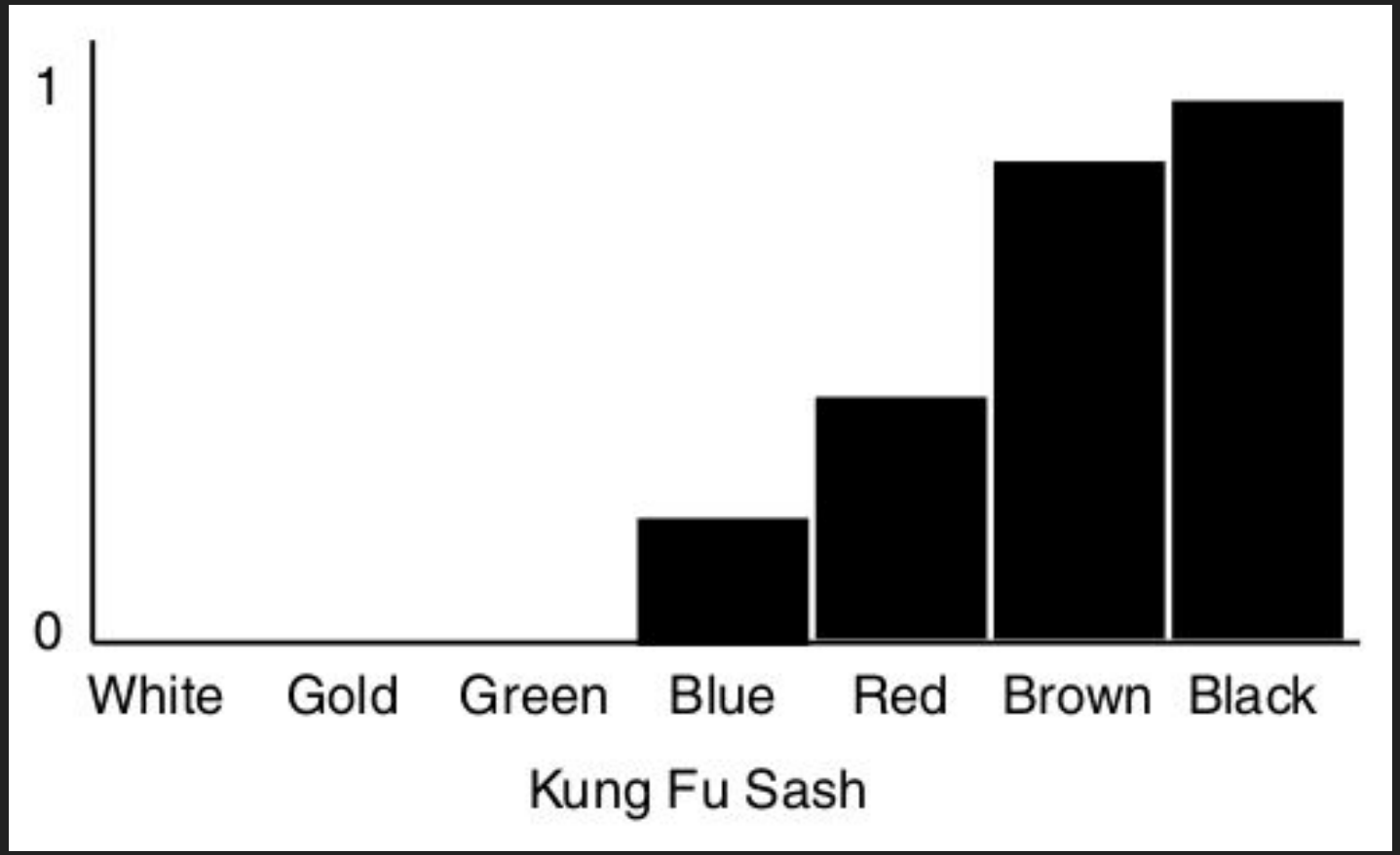
- Estende a noção de predicado lógico dando um valor a ele
  - Personagem pode estar faminto com um valor de 0.9
- Ao invés de pertencer ou não a um conjunto, tudo pode parcialmente pertencer a um conjunto
  - Com algumas coisas pertencendo mais que as outras
- Conjuntos *fuzzy*
  - Números são chamados de graus de pertencimento

Lógica *Fuzzy*

- Se dois conjuntos *fuzzy* são mutuamente exclusivos, ambos devem somar 1
  - O mesmo vale para *n* conjuntos



Fonte: Artificial Intelligence for Games



Fonte: Artificial Intelligence for Games

Processos de Markov

Processos de *Markov*

➔ Consideremos a segurança de 4 posições para um *sniper*

$$V = \begin{bmatrix} 1.0 \\ 0.5 \\ 1.0 \\ 1.5 \end{bmatrix}$$

Processos de *Markov*

➔ Matriz de transição se um tiro for dado

$$M = \begin{bmatrix} 0.1 & 0.3 & 0.3 & 0.3 \\ 0.0 & 0.8 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.8 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.8 \end{bmatrix}$$

Processos de *Markov*

➔ Vetor resultante

$$V = \begin{bmatrix} 0.1 \\ 0.7 \\ 1.1 \\ 1.5 \end{bmatrix}$$

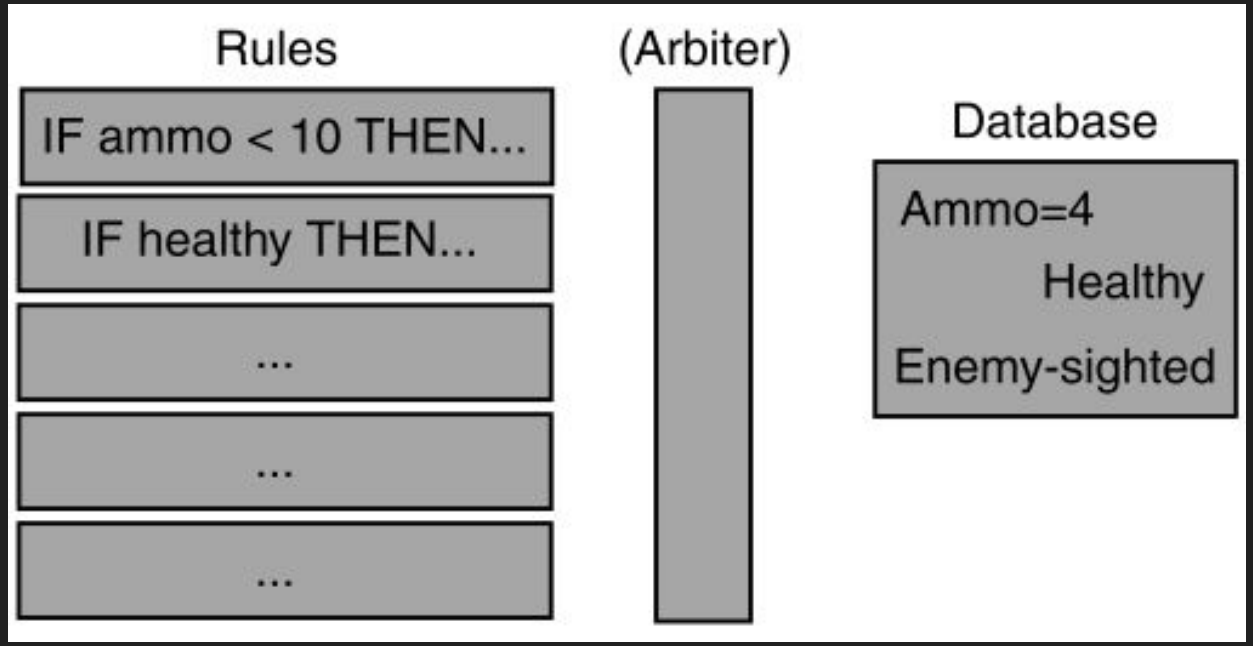
Processos de *Markov*

➔ Matriz de transição se não for dado um tiro

$$M = \begin{bmatrix} 1.0 & 0.1 & 0.1 & 0.1 \\ 0.1 & 1.0 & 0.1 & 0.1 \\ 0.1 & 0.1 & 1.0 & 0.1 \\ 0.1 & 0.1 & 0.1 & 1.0 \end{bmatrix}$$

Sistema Baseado em Regras

Sistema Baseado em Regras



Aprendizado

Aprendizado

- Tópico “quente” em jogos
- Pode ser *online* ou *offline*
- Muito complexo e pouco usado no mercado
- Mas muito pesquisado
- Veremos algumas técnicas de aprendizado de parâmetros

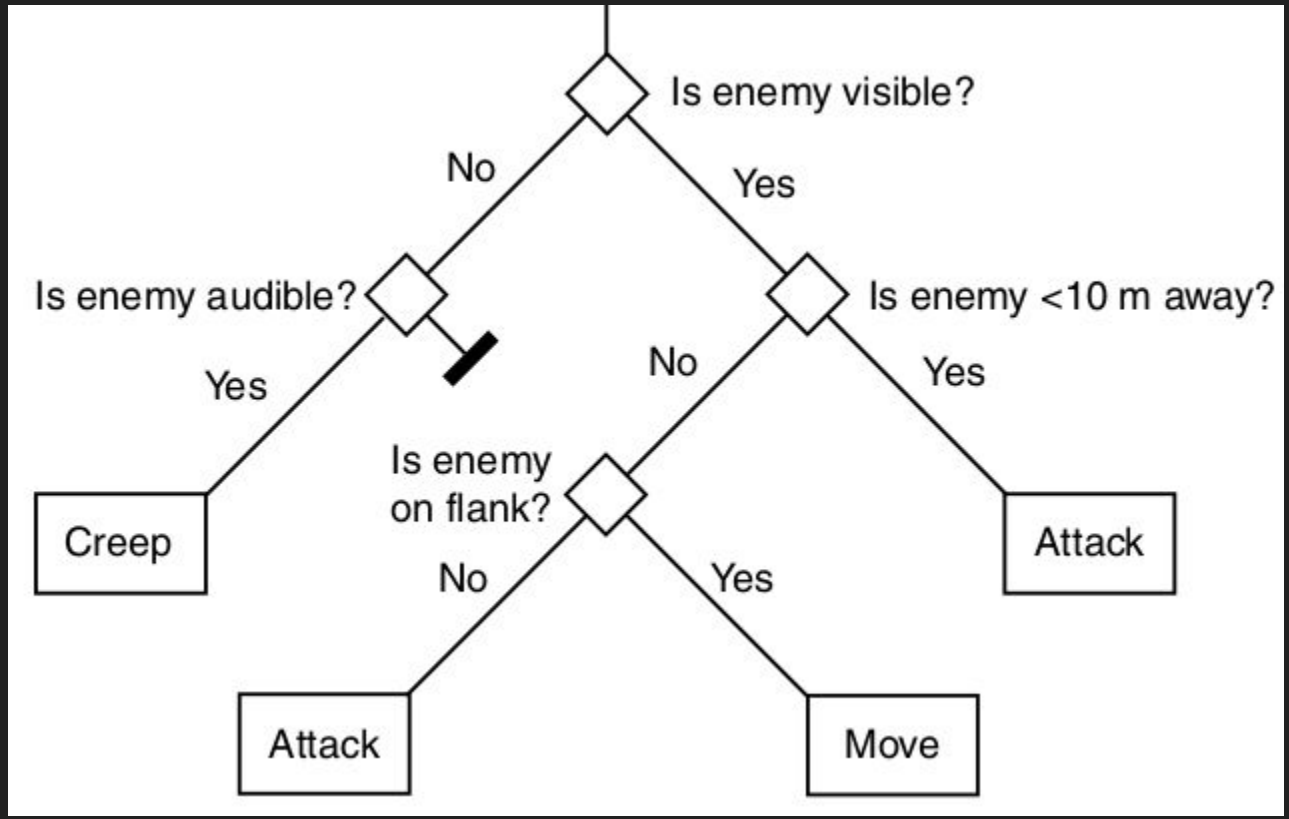
Hill Climbing

Annealing

AG

Árvores de Decisão

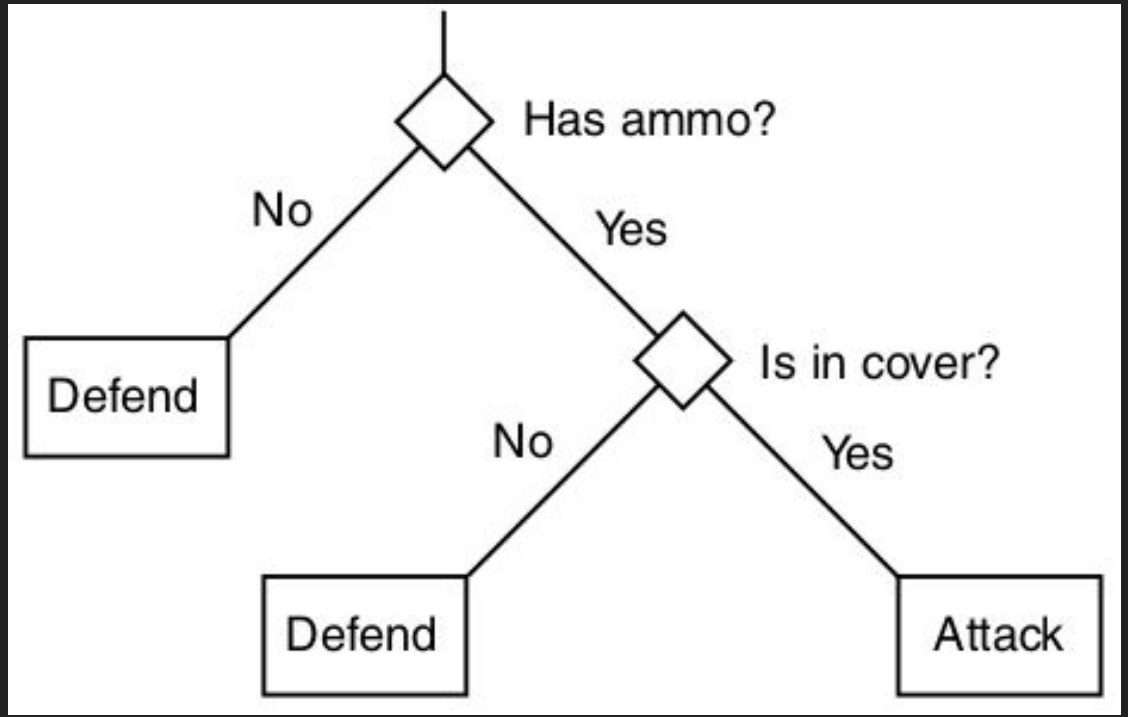
Árvores de Decisão



Árvores de Decisão

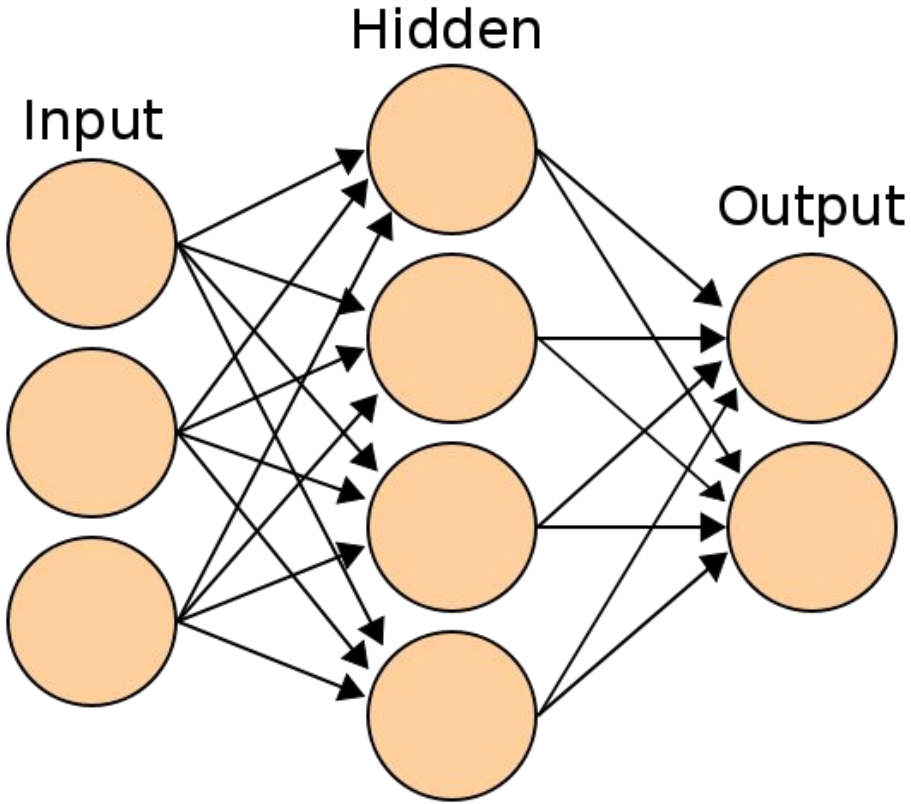
Healthy	In Cover	With Ammo	Attack
Hurt	In Cover	With Ammo	Attack
Healthy	In Cover	Empty	Defend
Hurt	In Cover	Empty	Defend
Hurt	Exposed	With Ammo	Defend

Árvores de Decisão



Redes Neurais

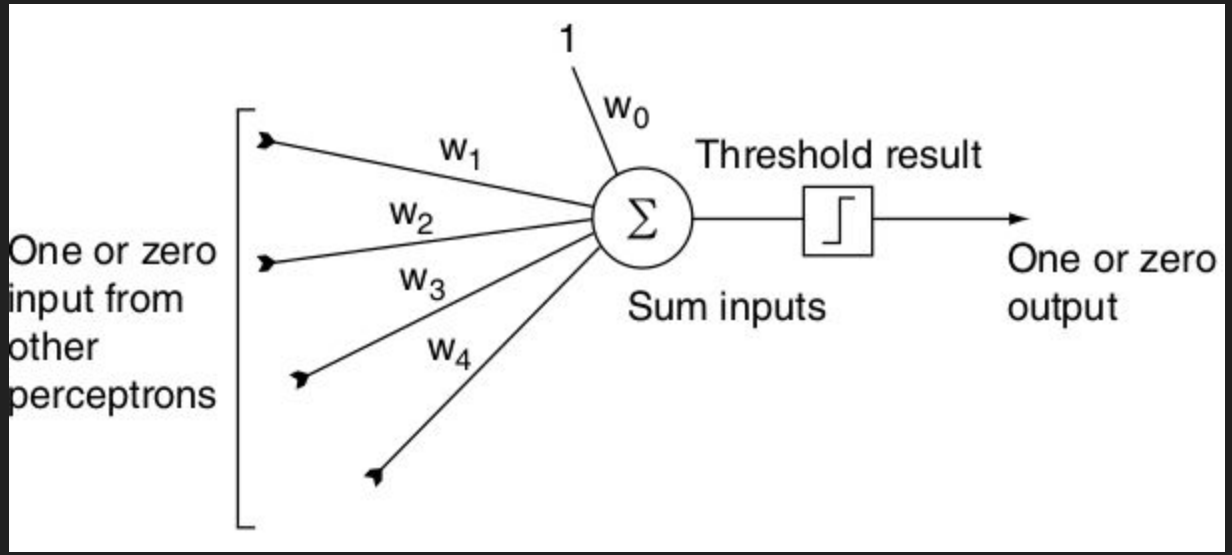
Redes Neurais

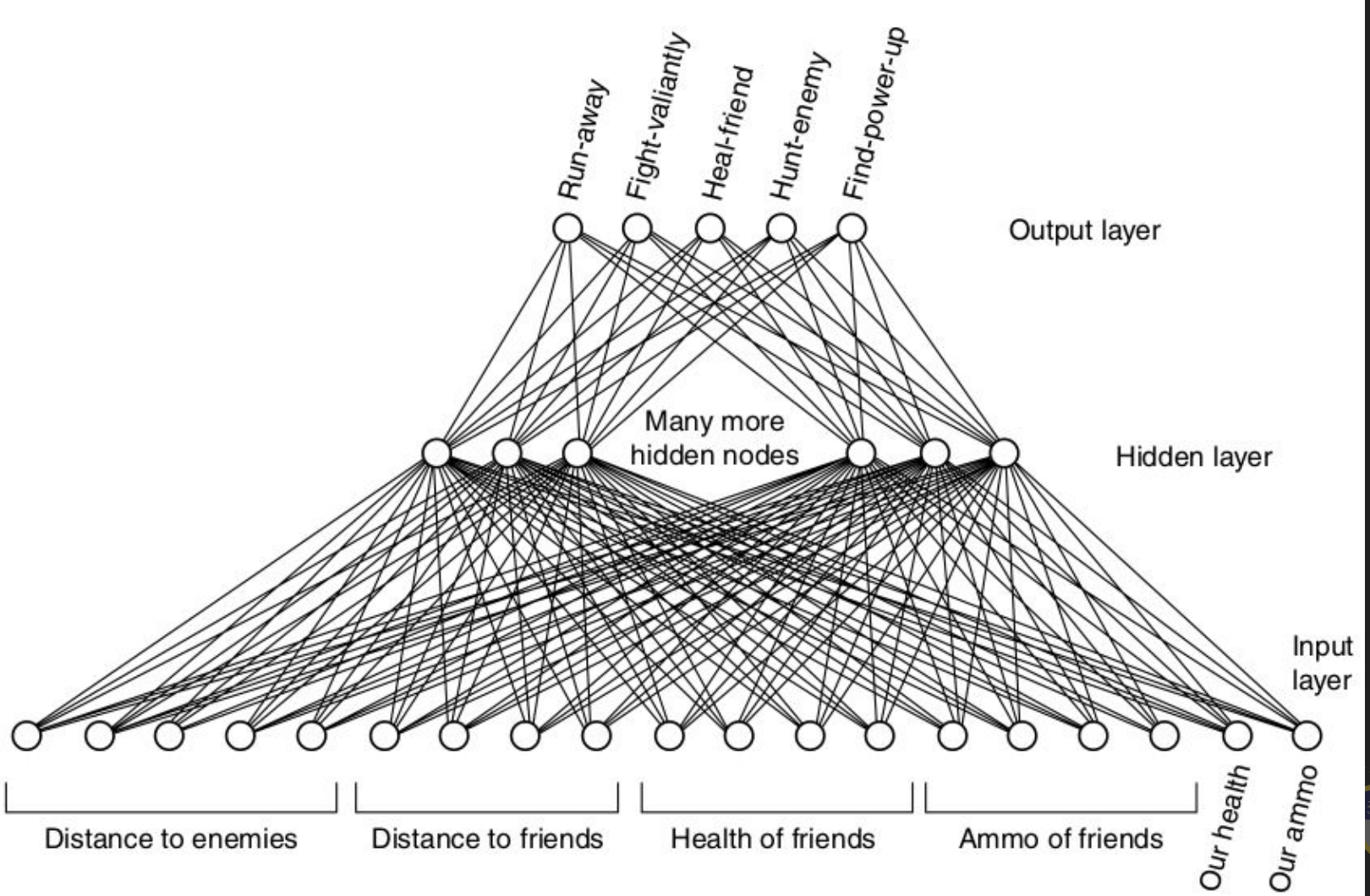


Redes Neurais

- Categorias principais
  - Acíclicas ou *feedforward*
  - Recorrentes
    - Perceptrons
    - Multi-layer perceptrons
    - RBN (Radial Basis Network)

Redes Neurais




Run-away
Fight-valiantly
Heal-friend
Hunt-enemy
Find-power-up
Output layer
Many more
hidden nodes
Hidden layer
Input
layer
Distance to enemies
Distance to friends
Health of friends
Ammo of friends
Our health
Our ammo

Redes Neurais

- *Deep learning*
  - Redes com muitas camadas
    - Uma treinada de cada vez
  - Podem modelar relacionamentos complexos não-lineares
  - Podem ser *feedforward* ou recorrentes
  - Recorrentes -> *Long Short Term Memory* (LSTM)
    - Revolucionou reconhecimento de fala

https://deepmind.com/blog/deepmind-and-blizzard-release-starcraft-ii-ai-research-environment/

Geração Procedural de Conteúdo

Geração Procedural de Conteúdo

- Criar dados algoritmicamente ao invés de manualmente
- Pode criar
  - Texturas
  - Modelos
  - Níveis
  - Itens
  - IAs
  - Audio
  - O que você conseguir!

Geração Procedural de Conteúdo

- Vantagens
  - Arquivos menores
  - Maiores conteúdos
  - Aleatoriedade
    - Jogabilidade menos previsível

- Fractais!

Rogue

https://en.wikipedia.org/wiki/Rogue_(video_game)#/media/File:Rogue_Screen_Shot_CAR.PNG

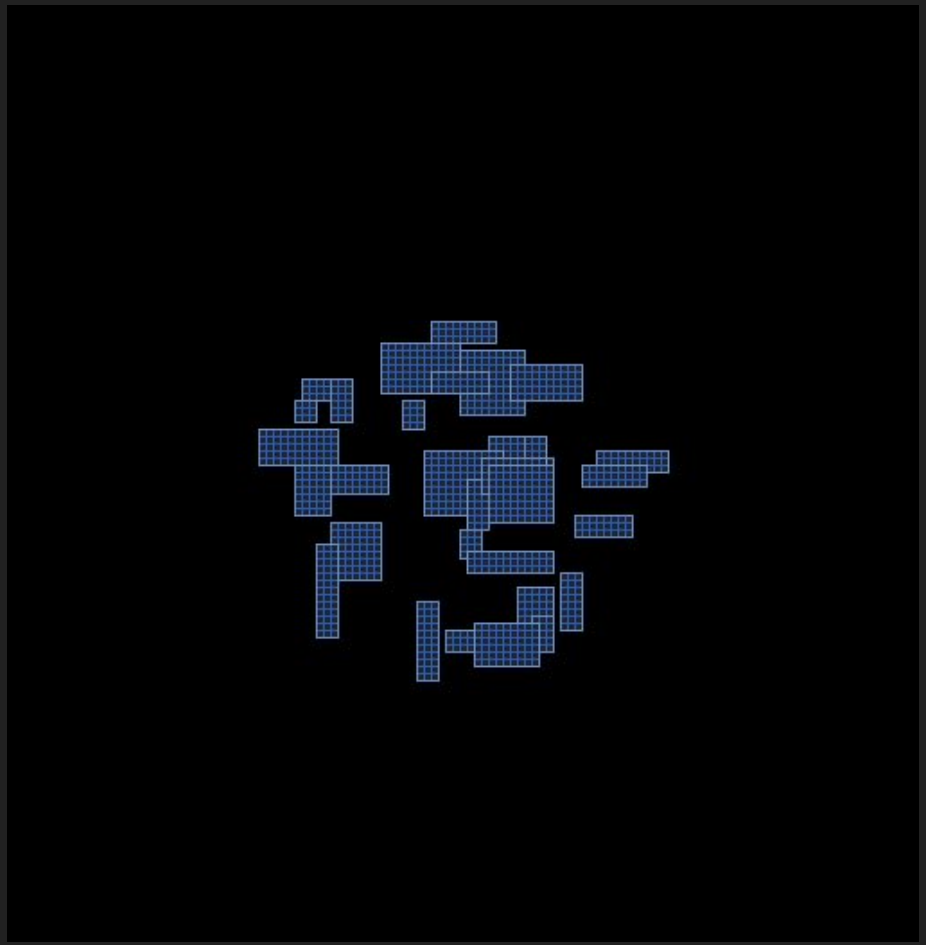
Geração Procedural de Conteúdo

- Dungeon generator
  1. Preencher mapa com terra
  2. Cria sala no centro do mapa
  3. Pegue uma parede de qualquer sala
  4. Decida qual característica construir
  5. Checar se existe espaço para construí-la pela parede
  6. Se sim, continuar. Se não, voltar para 3
  7. Adicionar a caracerística
  8. Voltar a 3 até completar o *dungeon*

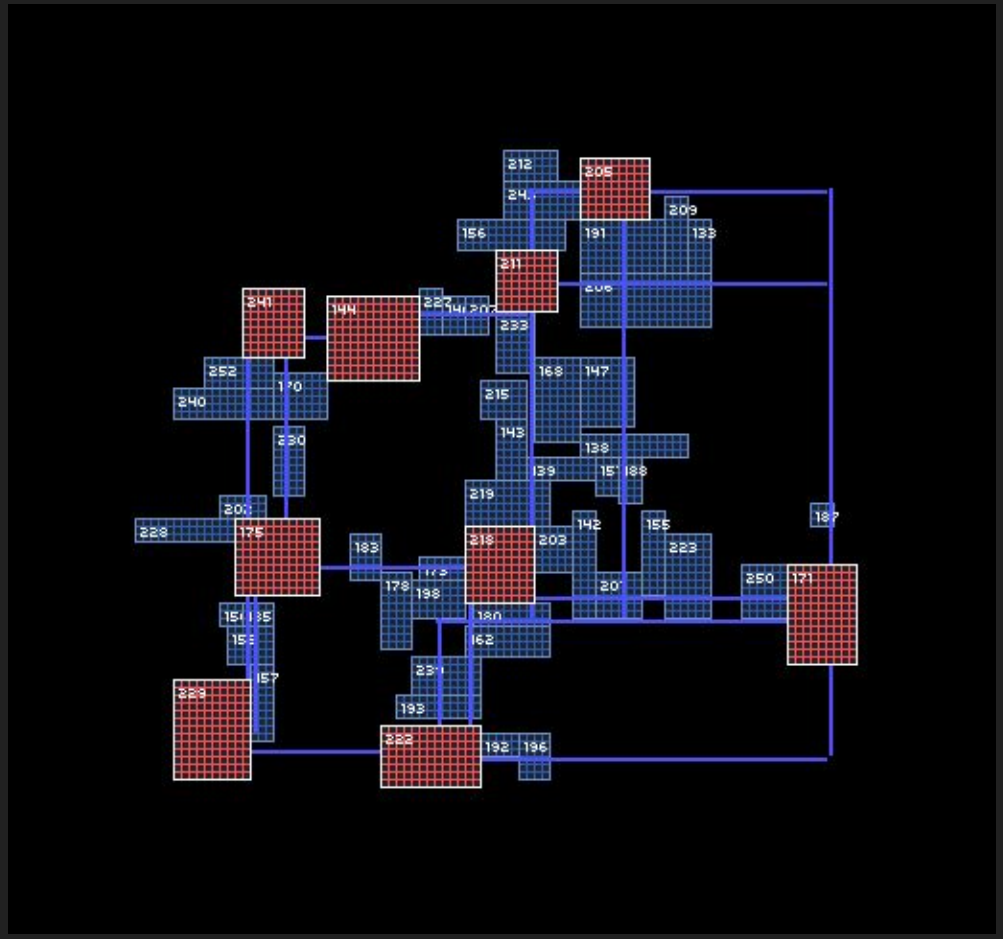
Geração Procedural de Conteúdo

➔ Dungeon generator (cont.)

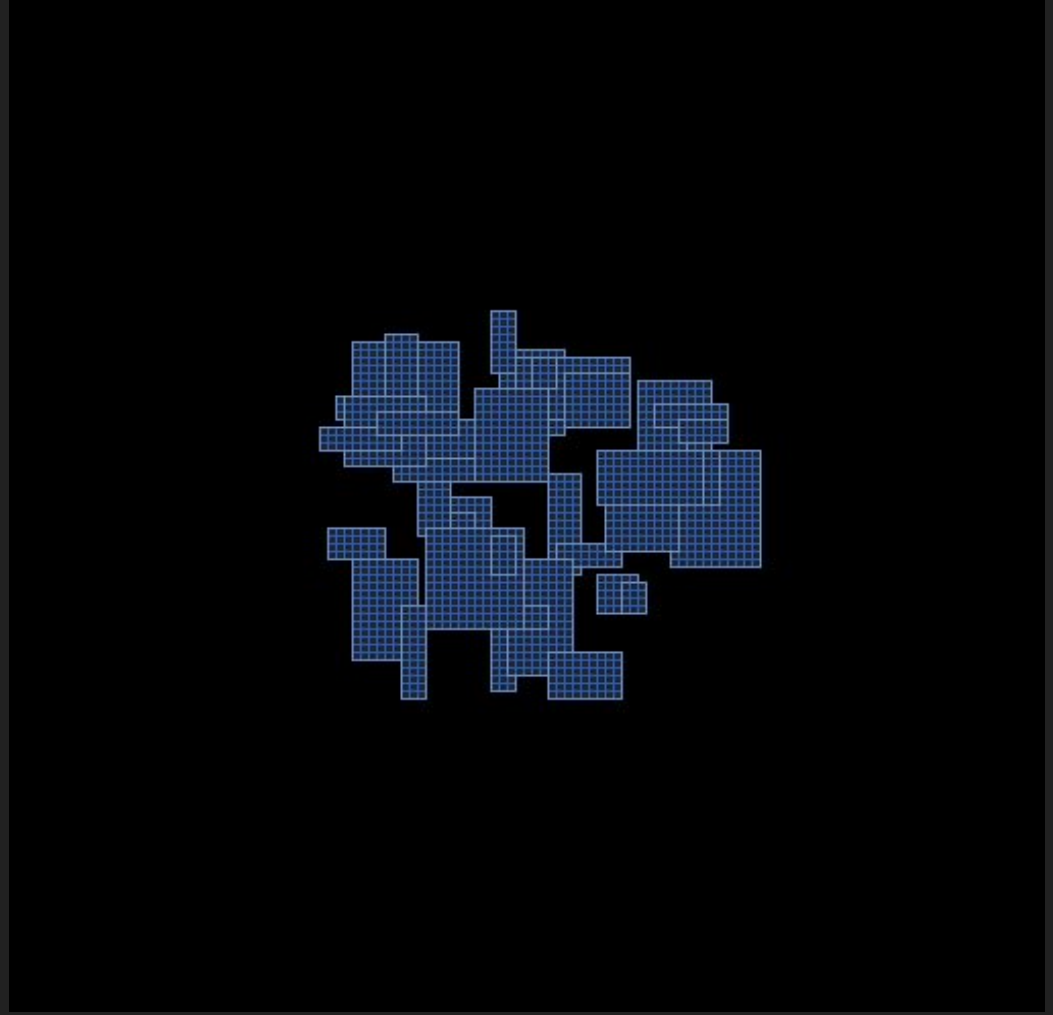
9. Adicionar escadas para cima e baixo aleatoriamente
10. Adicione pitadas de monstros e itens pelo *dungeon*



http://i.imgur.com/Spogz4z.gif



http://i.imgur.com/Y7E7Egn.png



http://i.imgur.com/GKO8EUG.gif

Geração Procedural de Conteúdo

- Monstros e suas partes em Spore
- Dificuldade em Left 4 Dead 2 (The Director)
- Itens em Borderlands 1 e 2
- Dungeons em Rogue Legacy, Hero Siege, Spelunky, Diablo
- Mundos do Minecraft
- Mundo e História em Dwarf Fortress
- .KKrieger, FPS de 96 Kb
- Árvore familiar em Crusader Kings II
- Muitos outros exemplos

Pesquisas em IA Aplicada a Jogos

Pesquisas em IA aplicada a jogos

- Área bem vasta
- Congressos especializados nisso (AIIDE)
  - http://www.aiide.org/
- Competições
  - http://ww12.platformersai.com/
  - https://aibirds.org
  - http://www.starcraftai.com/wiki/Main_Page

Pesquisas em IA aplicada a jogos

- Até *youtubers* pesquisam!
  - https://www.youtube.com/watch?v=qv6UVOQ0F44

Pesquisas em IA aplicada a jogos

- No ICMC
  - http://www.lucasnferreira.com/papers/2014/ace-edaab.pdf
  - http://www.lucasnferreira.com/papers/2014/gecco-mario.pdf
  - http://www.lucasnferreira.com/papers/2014/gecco-boss.pdf
  - http://www.lucasnferreira.com/papers/2014/cig-evoab.pdf

Pesquisas em IA aplicada a jogos

- De parceiros (Levi - UFV)
  - http://www.dpi.ufv.br/~lelis/papers/2015/reisLG15.pdf
  - http://www.dpi.ufv.br/~lelis/papers/2015/marinoRL15.pdf
- “O Cara”
  - http://julian.togelius.com/

Obrigado!

Referências

1. http://www.gamasutra.com/blogs/AAdonaac/20150903/252889/Procedural_Dungeon_Generation_Algorithm.php
2. http://www.roguebasin.com/index.php?title=Dungeon-Building_Algorithm
3. https://en.wikipedia.org/wiki/Artificial_intelligence
4. https://en.wikipedia.org/wiki/A*_search_algorithm
5. https://en.wikipedia.org/wiki/Procedural_generation
6. http://www.gamasutra.com/view/feature/174311/procedural_content_generation_.php
7. http://www.policyalmanac.org/games/aStarTutorial.htm
8. https://qiao.github.io/PathFinding.js/visual/
9. http://pcg.wikidot.com/major-pcg-games
10. http://www.gamasutra.com/view/news/262869/7_uses_of_procedural_generation_that_all_developers_should_study.php
11. https://en.wikipedia.org/wiki/Navigation_mesh
12. https://courses.nucl.ai/topic/pmgai-core-concept/
13. https://www.youtube.com/watch?v=Nt1XmiDwxhY&index=2&list=PLokhY9fbx05eq8SvcNOxYRquYMzMjF9Ok
14. https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-1
15. ARTIFICIAL INTELLIGENCE FOR GAMES. Millington, I; Funge, J. Burlington, MA : Elsevier Morgan Kaufmann, 2009. 2ª ed.