



Desenvolvimento de jogos digitais

LINGUAGENS E ALGORITMOS DE BUSCA



SUMÁRIO

- 1 Linguagens de programação para jogos
- 2 Busca de caminhos



SUMÁRIO

- 1 Linguagens de programação para jogos
- 2 Busca de caminhos



LINGUAGENS DE PROG. PARA JOGOS



- ❶ C++
- ❷ Java
- ❸ Linguagens de script



LINGUAGENS DE PROG. PARA JOGOS



C++

- ❶ Vantagens: alto desempenho, características de alto nível, herança de C, muitas bibliotecas disponíveis
- ❷ Desvantagens: baixo nível, complexidade, carência de recursos, compilação lenta



LINGUAGENS DE PROG. PARA JOGOS



Java

- ❶ Vantagens: alto nível, serialização de objetos, portabilidade, bibliotecas disponíveis, curva de aprendizado
- ❷ Desvantagens: pouco controle sobre o funcionamento no hardware, desempenho, gerenciamento de memória



LINGUAGENS DE PROG. PARA JOGOS



Linguagens de script

- ❶ Vantagens: facilidade de desenvolvimento, tempo de compilação, o código se torna um recurso, recursos disponíveis
- ❷ Desvantagens: desempenho, suporte de ferramentas, dificuldade para encontrar erros no código, interface com o restante do jogo



LINGUAGENS DE PROG. PARA JOGOS



Linguagens de script - LUA

- ❶ Desenvolvida no Brasil em 1993
- ❷ Considerada a linguagem de script mais usada em games (2009)
- ❸ Leve: 245KB (Linux x64)

```
print "Olá, Mundo!"
```

```
prompt> lua -e "print(2^0.5)"
```







SUMÁRIO

- 1 Linguagens de programação para jogos
- 2 Busca de caminhos



BUSCA DE CAMINHOS (PATHFINDING)



-  Movimentar-se de um lugar a outro é uma característica elementar de uma agente.
-  Caminho deve ser razoável
-  Obstáculos
-  Importante balancear:
Precisão x Desempenho

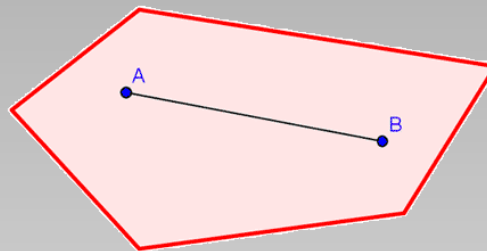
-  Algoritmo A*



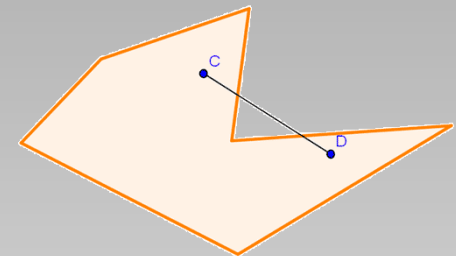
BUSCA DE CAMINHOS (PATHFINDING)



- 🍄 O espaço de busca pode ser muito grande.
- 🍄 Grades: conjunto de células, transitáveis e intransitáveis.
- 🍄 Waypoints: marcações no ambiente.
- 🍄 Malhas de navegação: conjunto de polígonos convexos que descreve as superfícies caminháveis do mundo.



Polígono convexo

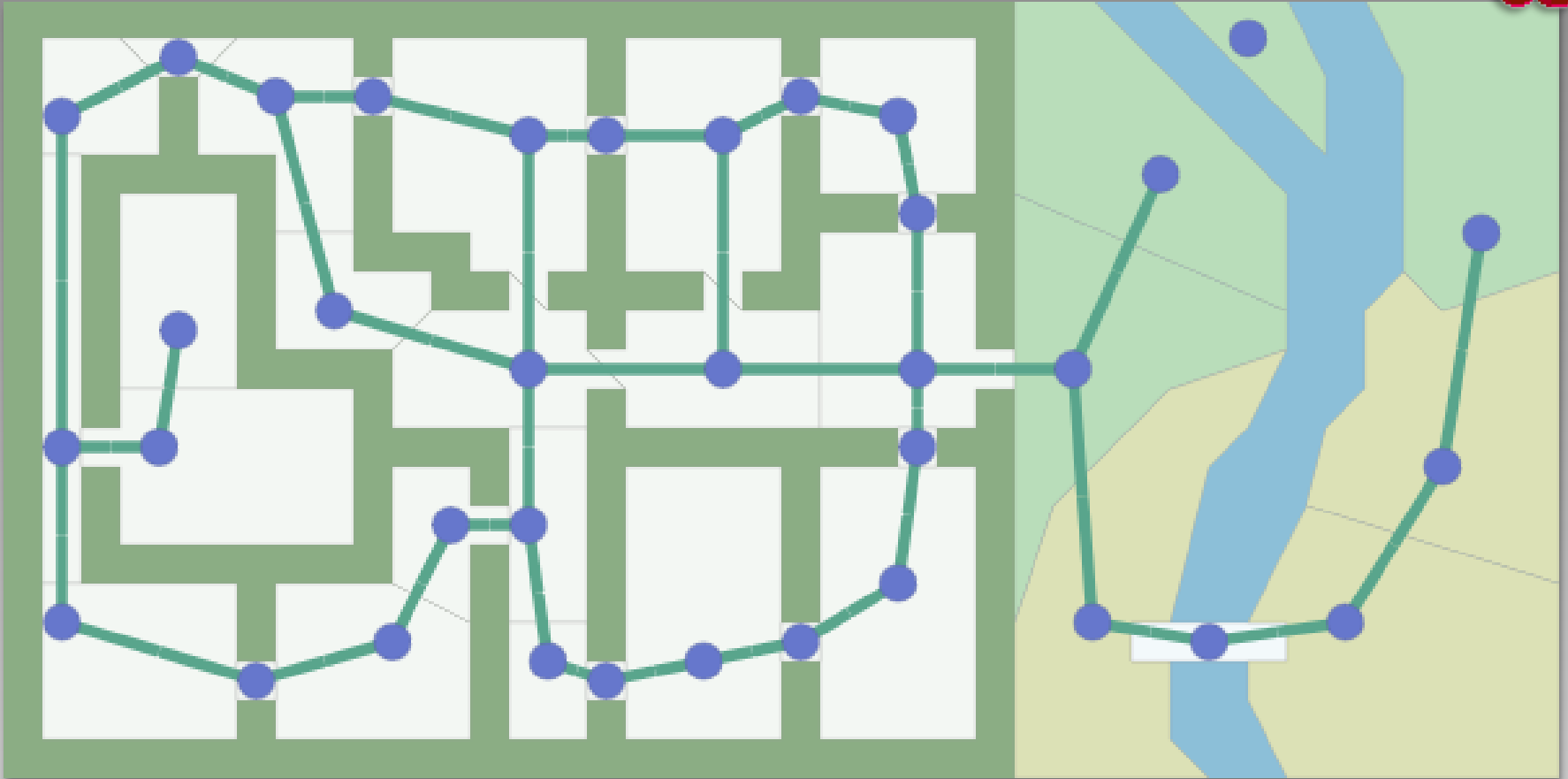


Polígono não-convexo

Fonte: <http://www.matematica.pt/faq/poligono-convexo.php>



BUSCA DE CAMINHOS (*PATHFINDING*)

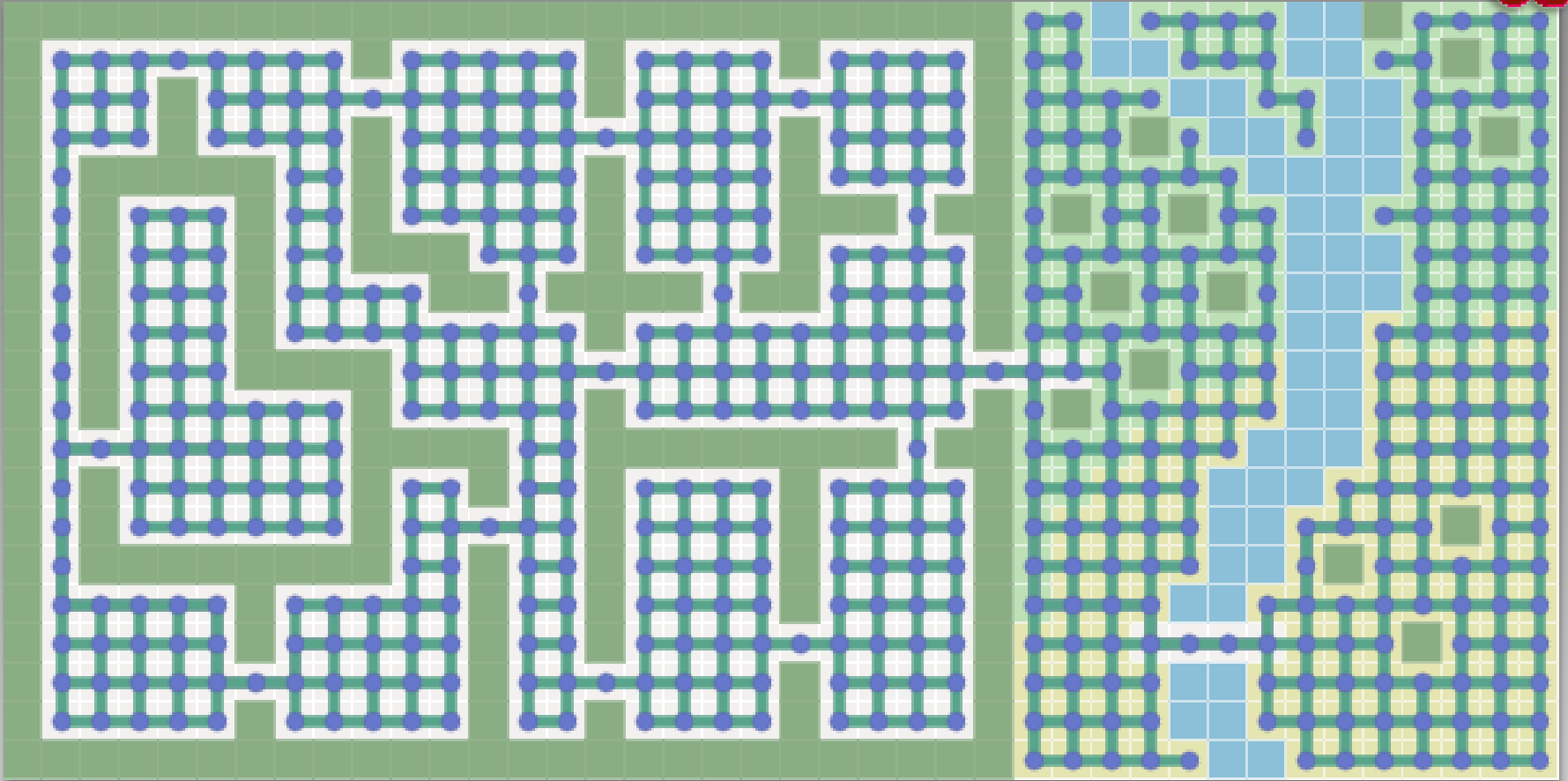


Fonte: <http://www.redblobgames.com/pathfinding/a-star/introduction.html>

Waypoints



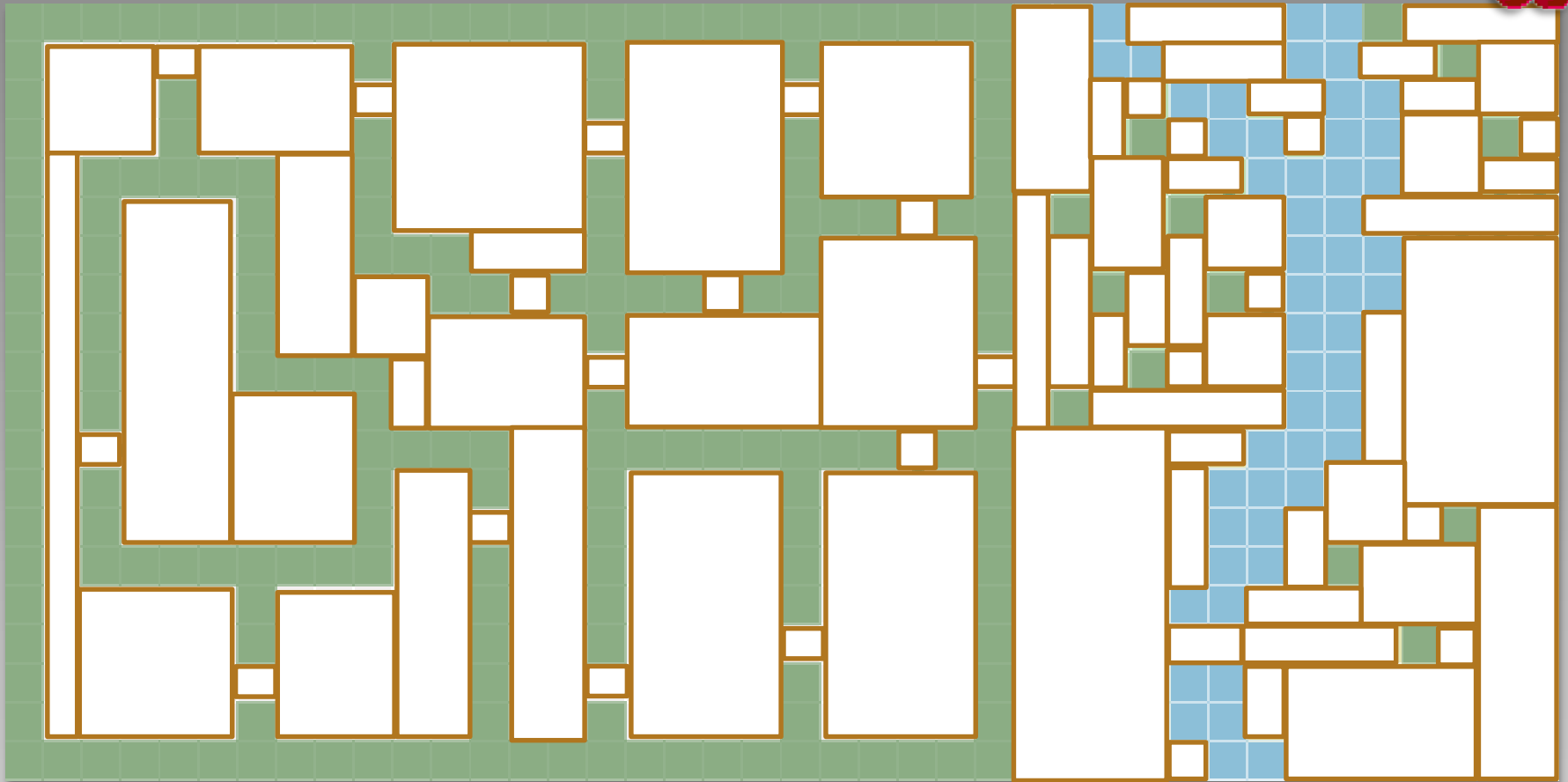
BUSCA DE CAMINHOS (PATHFINDING)



Fonte: <http://www.redblobgames.com/pathfinding/a-star/introduction.html>

Waypoints
Grades

BUSCA DE CAMINHOS (*PATHFINDING*)

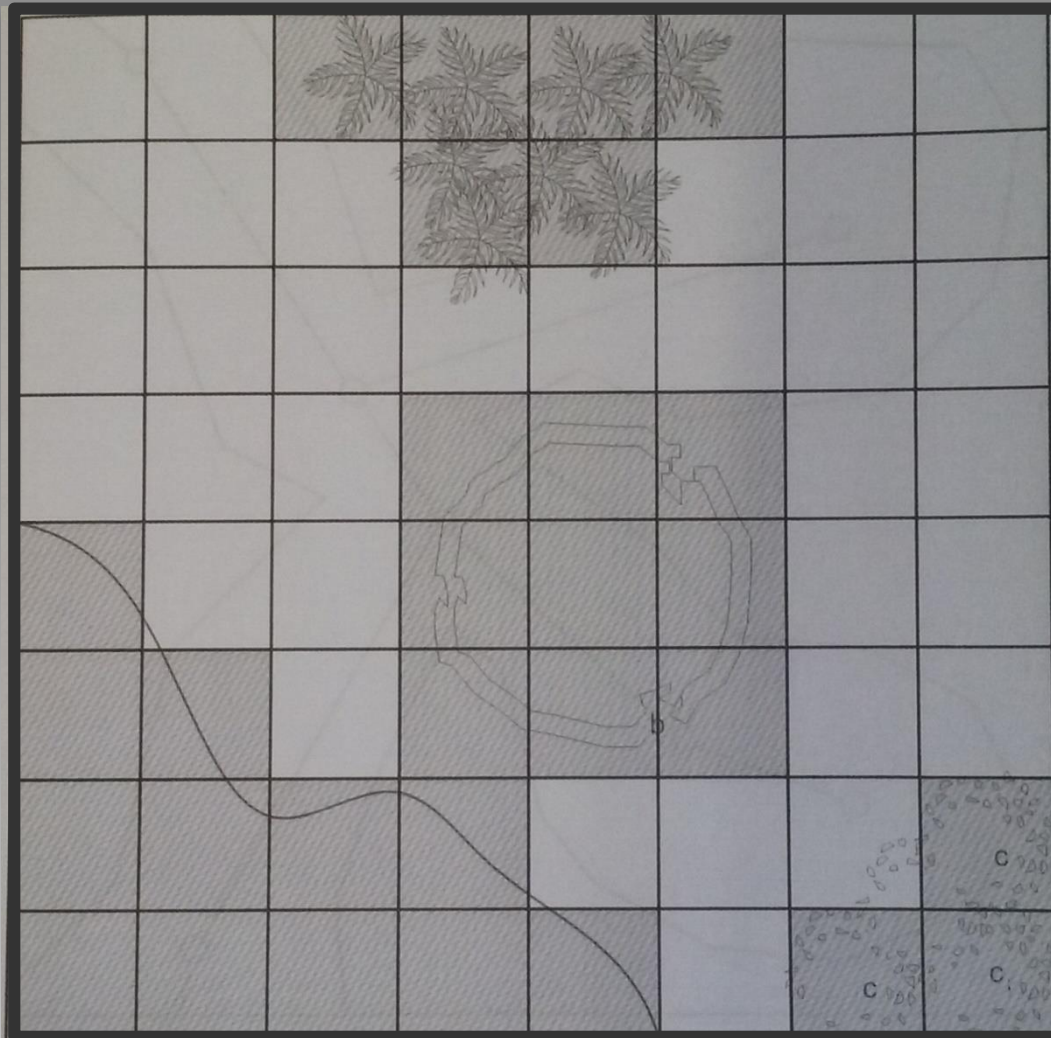


Fonte: <http://www.redblobgames.com/pathfinding/a-star/introduction.html>

Malha de
navegação



BUSCA DE CAMINHOS (PATHFINDING)



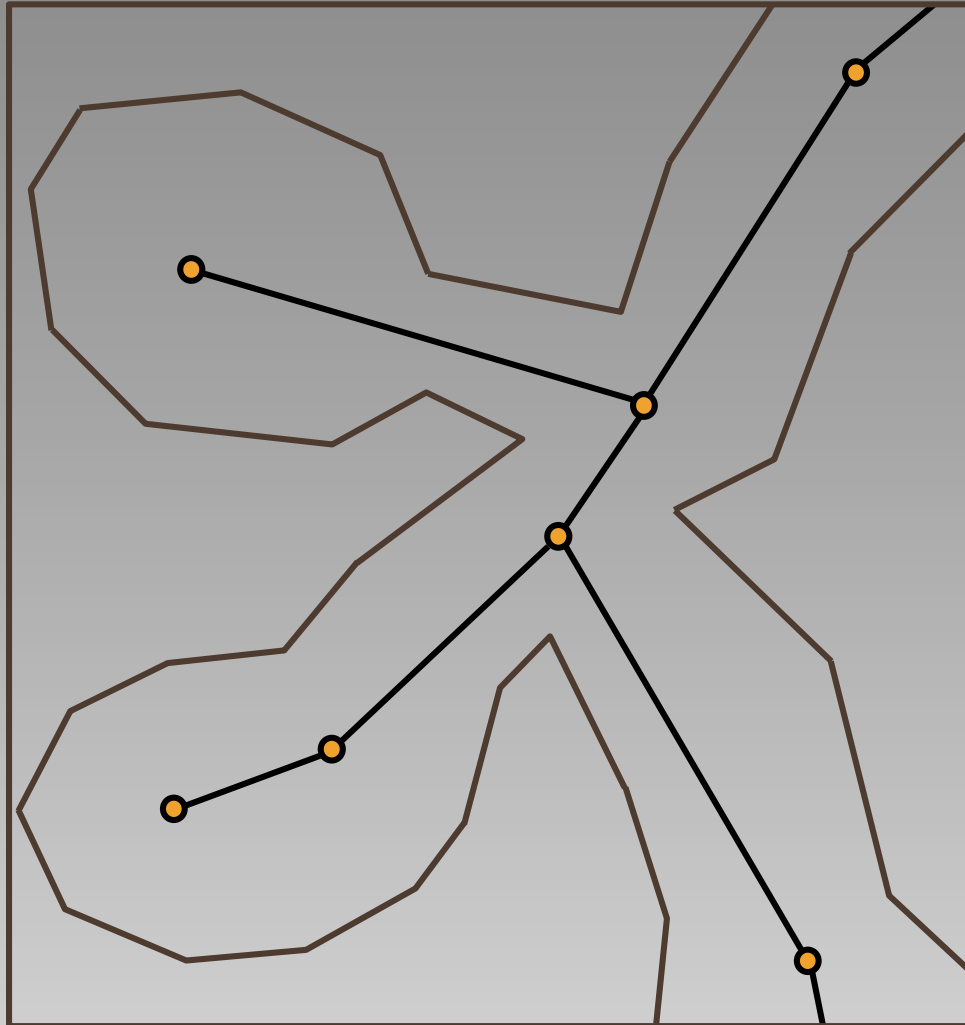
Nível de um game
representado com
Grade



Fonte: Introdução ao Desenvolvimento de Games. Vol2. Cengage Learning.



BUSCA DE CAMINHOS (PATHFINDING)



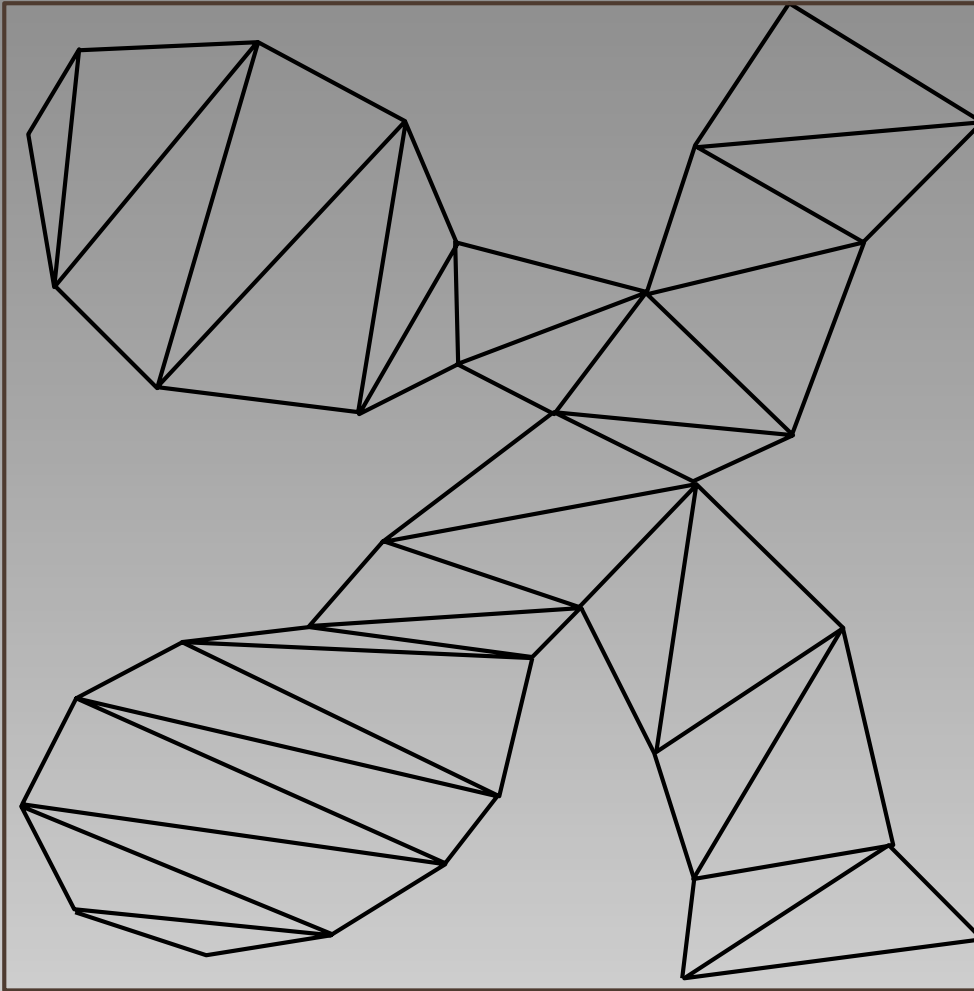
Nível de um game
representado com
Grafo de Waypoints



Fonte original: Introdução ao Desenvolvimento de Games. Vol2. Cengage Learning.



BUSCA DE CAMINHOS (PATHFINDING)



Nível de um game
representado com
Malha de navegação



Fonte original: Introdução ao Desenvolvimento de Games. Vol2. Cengage Learning.



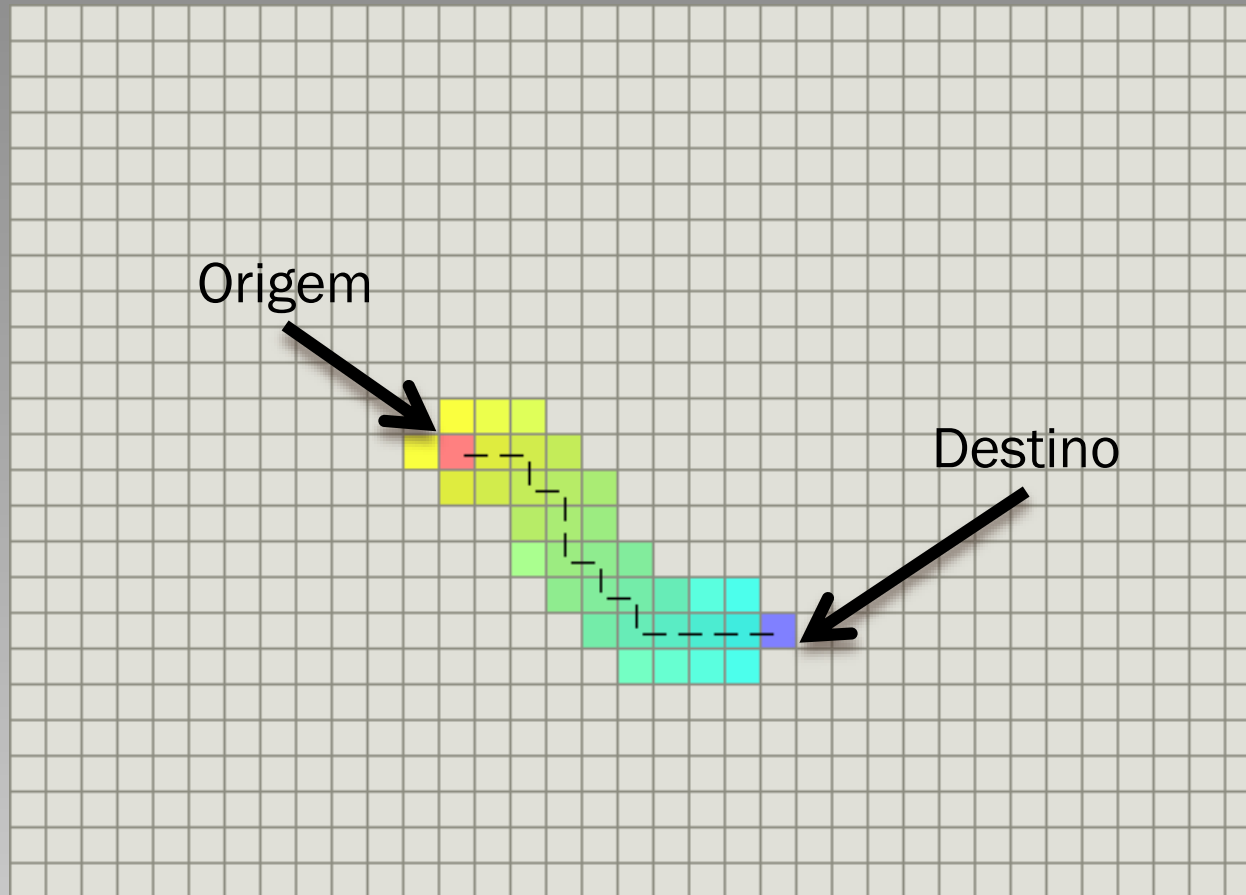
PATHFINDING - ALGORITMO A*



- 🍄 Escolha mais popular para pathfinding.
- 🍄 Encontra o melhor caminho entre dois pontos.
- 🍄 Variação do algoritmo Dijkstra incluindo uma função heurística.



PATHFINDING - ALGORITMO A*



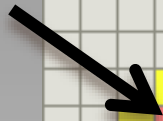
Fonte: <http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html>



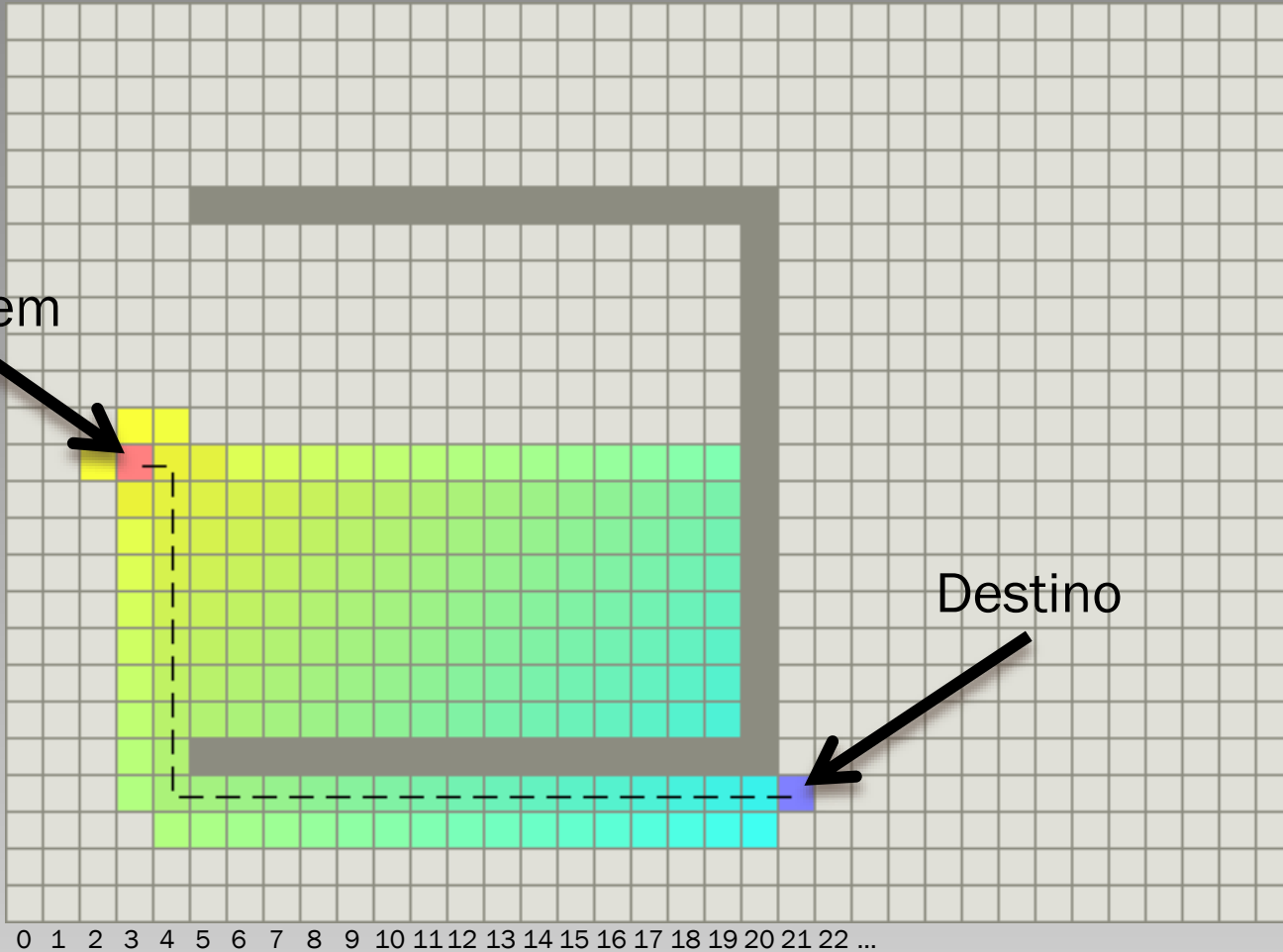
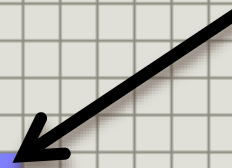
PATHFINDING - ALGORITMO A*



Origem



Destino



Fonte: <http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html>



PATHFINDING - ALGORITMO A*

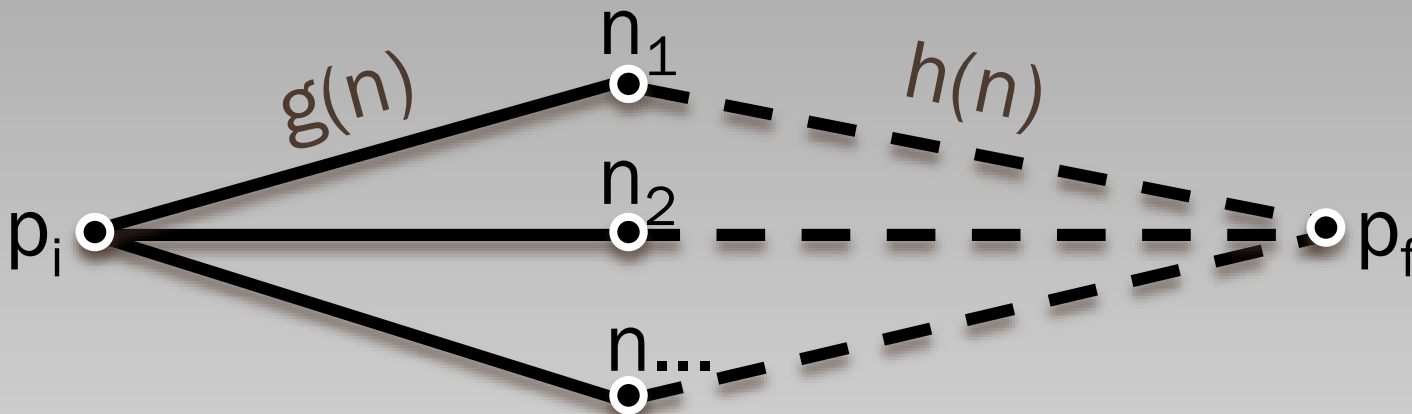


p_i = ponto inicial ; p_f = ponto final

🍄 $g(n)$ = custo exato do caminho de p_i até n_i

🍄 $h(n)$ = custo estimado do caminho de n_i até p_f

Loop $\rightarrow f(n) = g(n) + h(n)$



PATHFINDING - ALGORITMO A*



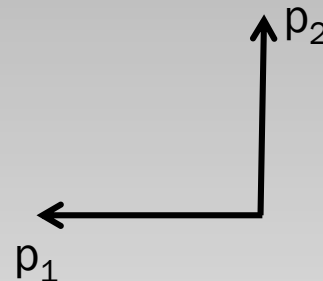
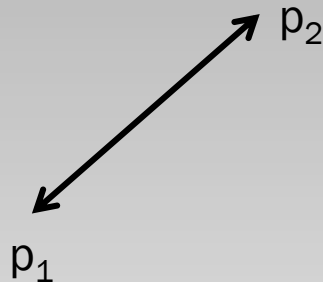
🍄 Funções heurísticas comuns:

🍄 Distância Euclidiana

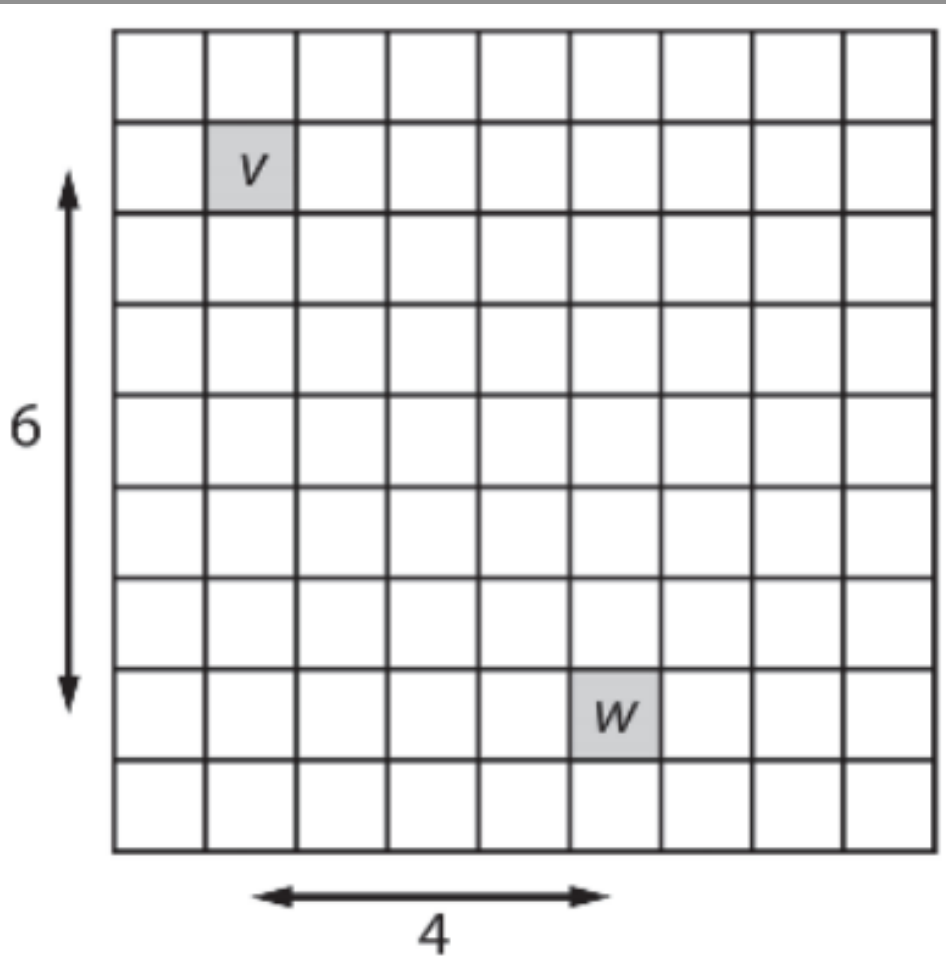
$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

🍄 Distância de Manhattan

$$d((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$$



PATHFINDING - ALGORITMO A*



Distância de
Manhattan

$$\begin{aligned} &= |v_x - w_x| + |v_y - w_y| \\ &= |2 - 6| + |2 - 8| \\ &= 10 \end{aligned}$$



Fonte: Programming Game AI by Example, Mat Buckland, 2004



PATHFINDING - ALGORITMO A*



Algoritmo resumido

- 1) Cria lista P
- 2) Adiciona nodo inicial S à lista P
- 3) Até que o primeiro caminho de P termine no nodo final ou P esteja vazia
- 4) Extraia o primeiro caminho de P
- 5) Extenda o primeiro caminho 1 passo para todos seus vizinhos sem loops, cujos custos sejam menores que outros caminhos com mesmo nodo final
- 6) Adicione cada novo caminho a P, ordenado por menor custo total (distância percorrida + heurística)
- 7) Se achou o nodo final → sucesso. Senão → Não existe caminho.



Fonte: <http://www.inf.ufrgs.br/~rcpinto/facul/inf05515/Complexidade%20do%20Algoritmo%20A.htm>



PATHFINDING - ALGORITMO A*



Algoritmo detalhado contextualizado

1) Adicione o quadrado inicial à **lista aberta**.

2) Repita:

a. Procure o quadrado que tenha o menor custo de F na **lista aberta**. → **quadrado corrente**

b. Mova-o para a **lista fechada**.

c. Para cada um dos **4 quadrados adjacentes** a este quadrado corrente.

Se não é passável ou se estiver na **lista fechada**, ignore. Caso contrário faça:

c.1) Se não estiver na **lista aberta**, acrescente-o à **lista aberta**. Faça o **quadrado corrente** o pai deste quadrado. Grave os custos F, G, e H do quadrado.

c.2) Se já estiver na **lista aberta**, conferir se este caminho para aquele quadrado é melhor, usando custo G como medida. **Um valor G mais baixo mostra que este é um caminho melhor**. Nesse caso, mude o **pai do quadrado para o quadrado corrente**, e recalcule os valores de G e F do quadrado. Se você está mantendo sua lista aberta ordenada por F, você pode precisar reordenar a lista para corresponder a mudança.

3) Parar quando:

a) **Acrescentar o quadrado alvo à lista fechada** o que determina que o caminho foi achado, ou

b) **Não achar o quadrado alvo**, e a lista aberta está vazia. Neste caso, não há nenhum caminho.



PATHFINDING - ALGORITMO A*



>Implementação...

