Technische Universität Darmstadt

Seminar: Knowledge Engineering und Lernen in Spielen, SS06 bei Prof. Dr. Fürnkranz

Thema: A* Pathfinding

Daniel Sutlar

16.05.2006

Motivation

- Pathfinding ist ein grundlegendes Problem in Computerspielen
- schlechtes Pathfinding lässt die Spielcharaktere unintelligent und verwirrt erscheinen
- Spielspass und Motivation gehen verloren

Lösung bietet der A* Algorithmus

Überblick

1. Begriffsklärung
2. Funktionsweise des A* Pathfinding
3. Erweiterungen
4. Fazit
5. Demo am Spiel Remote Assault

Begriffsklärung

Pathfinding

- Wegsuche/Wegfindung
- schnellsten/kürzesten/optimalen Weg von A nach B finden
- Einsatzgebiet: u.a. Routenplaner und Computerspiele; Echtzeit-Strategiespielen und Ego-Shootern

- Age of Empires
- Command & Conquer
- Die Siedler
- Empire Earth
- WarCraft
- usw.

Begriffsklärung (2)

A* Pathfinding

- bekannteste und meist benutzte Pathfinding-Methode in der Spieleentwicklung
- A* Algorithmus; 1968 von Peter Hart, Nils Nillson und Bertram Raphael entwickelt
- „informierte Suche“, d.h. Anwendung einer Heuristik
- Knoten untersuchen, welche am wahrscheinlichsten zum Ziel führen

Funktionsweise

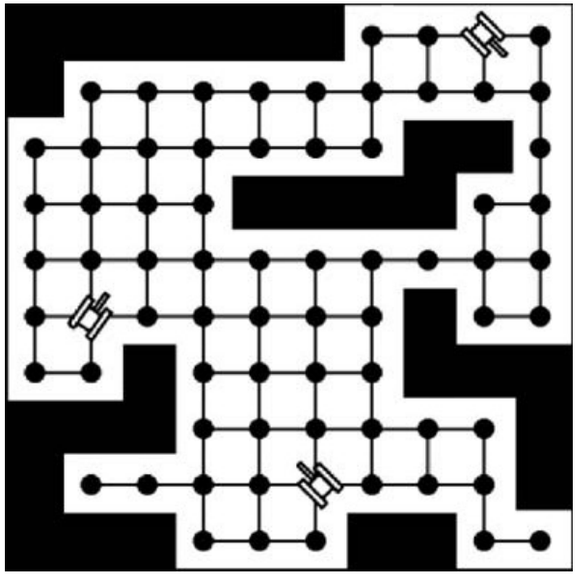
1. Festlegung des Suchbereiches

2. Anwendung des A* Algorithmus

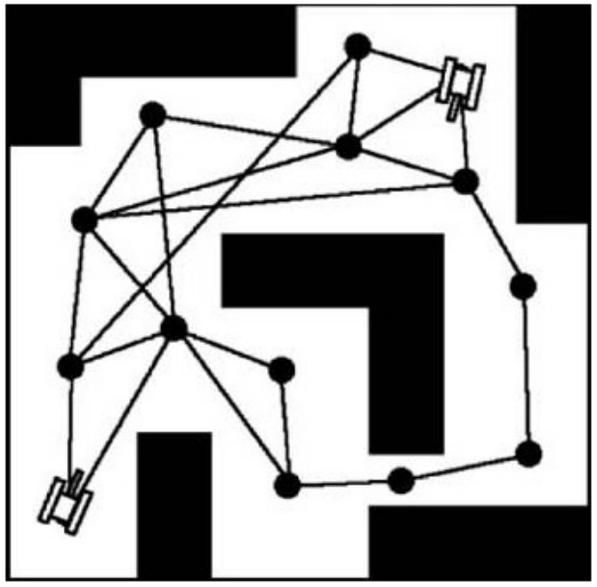
4. Erweiterungen

Festlegung des Suchbereiches

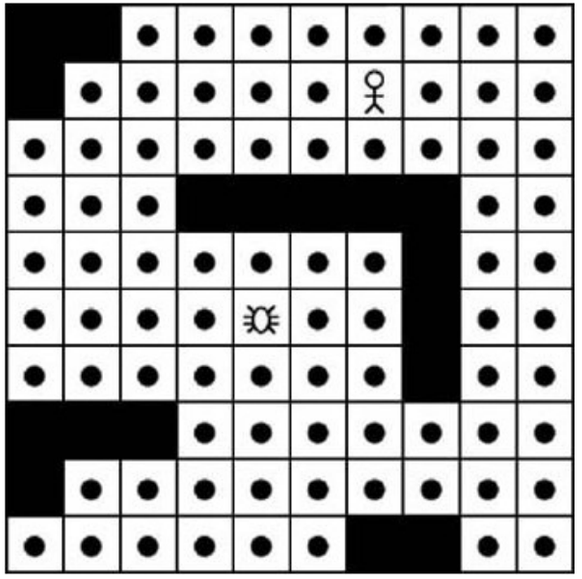
- Unterteilung des Suchbereichs/Spielwelt in einzelne angemessene Flächen
- ein Knoten pro Fläche
- Verbindungen zwischen Knoten



Simplyfing the search area



Continuous node placement



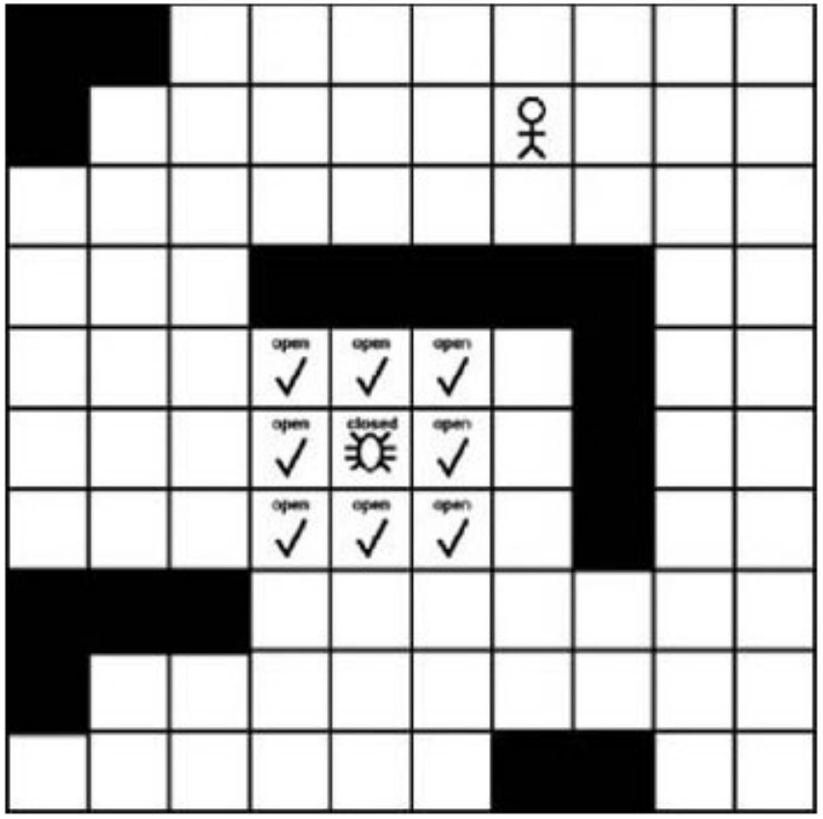
Tiled search area

Anwendung des A* Algorithmus

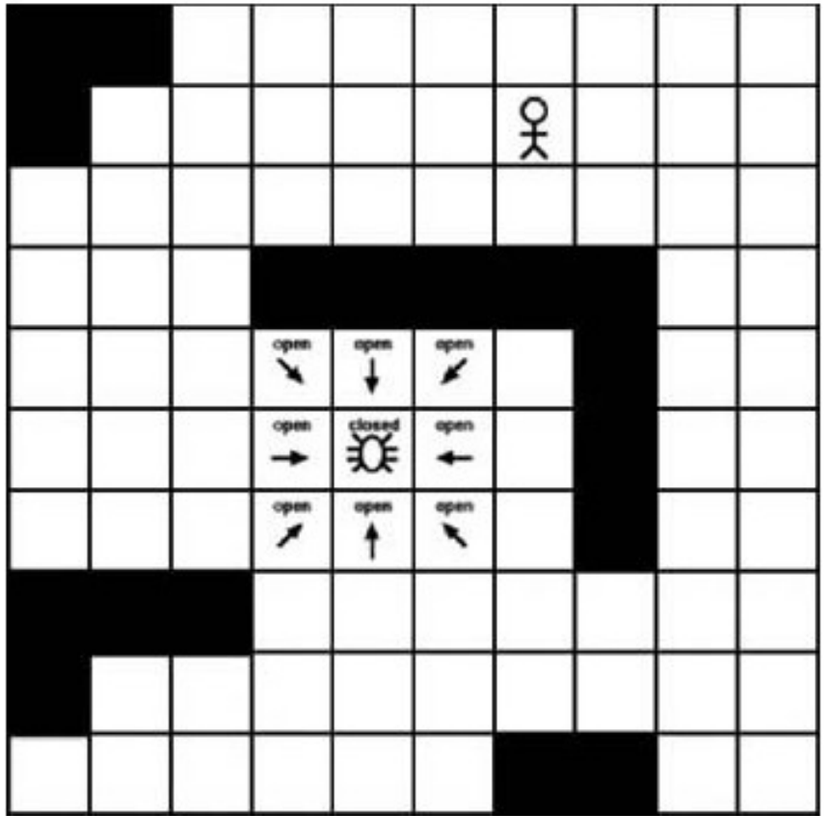
Suche Starten

- Startknoten und Endknoten definieren
- Vorgehensweise:
  1. Beginne mit Startpunkt A und füge ihn einer "offenen" Liste hinzu.
  2. Schaue Dir alle Quadrate an, die an das Startquadrat A grenzen, und ignoriere dabei Quadrate, welche nicht begehbares Terrain darstellen. Füge sie der o.g. Liste hinzu und merke für jedes dieser Quadrate das Startquadrat A als seinen Vorgänger.
  3. Wirf das Startquadrat A aus Deiner bisherigen „offenen Liste“ und füge es zu einer anderen, "geschlossenen" Liste von Quadraten hinzu
  4. Berechne Wegkosten für jedes neue Quadrat in der „offenen Liste“

Beispiel A* Algorithmus



Prüfung der Nachbarfelder



Verknüpfung zu Elternknoten

Pfadbewertung

- Wegkosten zur Bestimmung des kürzesten Pfades

Gesamt-Wegkosten =
Wegkosten vom Startpunkt + Wegkosten zum Endpunkt (geschätzt)

Wegkosten vom Startpunkt:

- Bewegungskosten vom Startpunkt zu einem gegebenen Quadrat unter Verwendung des dafür ermittelten Pfades
- jeder möglichen Bewegungsrichtung wird ein Kostenwert zugewiesen, z. B. Bewegung in horizontaler/vertikaler Richtung mit Kosten von 10, diagonaler Richtung Kosten von 14

Pfadbewertung 2

Wegkosten zum Endpunkt

- Abschätzung der Kosten mit Heuristik
- Kosten nicht überschätzen und keine negativen Kosten
- Bsp. für Heuristik:
  - Manhatten Methode (nur vertikale und horizontale Bewegung einkalkulieren; keine Beachtung von Hindernissen)

    h(n) = D * (abs(n.x-goal.x) + abs(n.y-goal.y))
  - Diagonal Abstand (diagonale Bewegung mit einrechnen)

    h(n) = D * max(abs(n.x-goal.x), abs(n.y-goal.y))
  - Euklidscher Abstand (vgl. Luftlinienentfernung)

    h(n) = D * sqrt((n.x-goal.x)^2 + (n.y-goal.y)^2)

Anwendung des A* Algorithmus - Fortsetzung

Fortsetzung der Suche:

- Wähle aus der offenen Liste das Quadrat mit dem niedrigsten Gesamt-Wegkosten aus
- Entferne es aus der offenen Liste und füge es der geschlossenen Liste hinzu
- Prüfe alle angrenzenden Quadrate und füge sie der offenen Liste hinzu, sofern sie:
  - kein Hindernis darstellen
  - sich nicht bereits in der offenen Liste befinden
  - sich nicht in der geschlossenen Liste befinden
- Trage für jedes dieser Quadrate das aktuelle Quadrat als Vorgängerquadrat ein

Anwendung des A* Algorithmus - Fortsetzung

- falls eines der benachbarten Quadrate sich bereits in der offenen Liste befindet, dann prüfe:
  - ob der Pfad vom aktuellen Quadrat zu ihm geringere Wegkosten vom Startpunkt aufweist, als der Pfad von seinem eingetragenen Vorgängerquadrat
  - falls ja, ändere sein Vorgängerquadrat auf das aktuelle Quadrat und berechne seine Werte neu

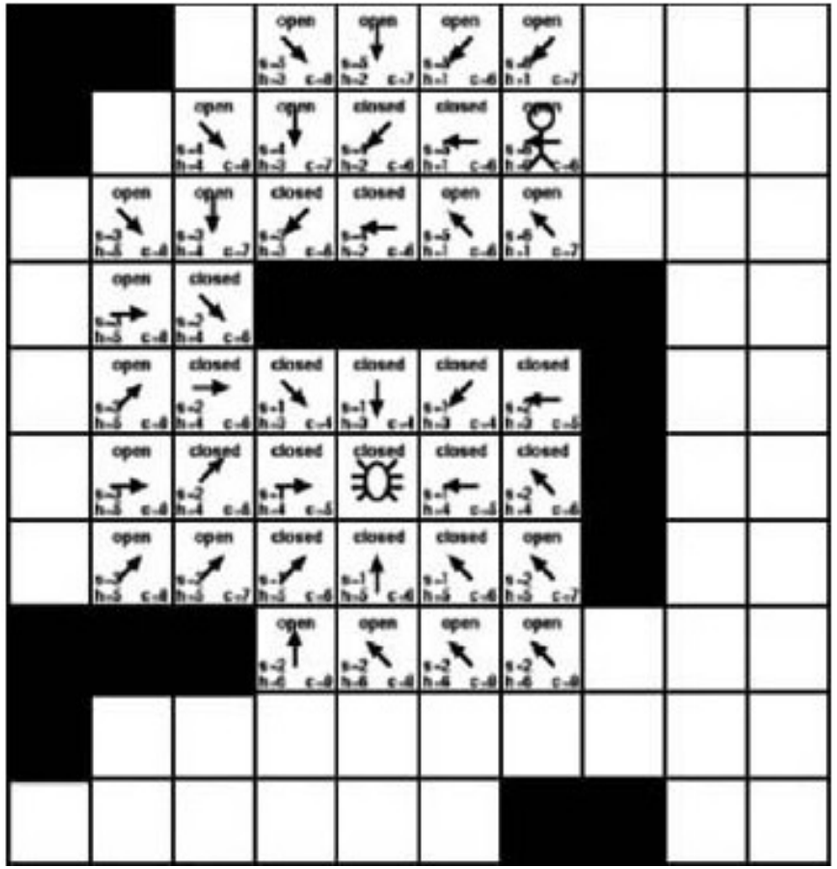
Beendigung der Suche

A* Methode kommt zum Abschluss, falls:

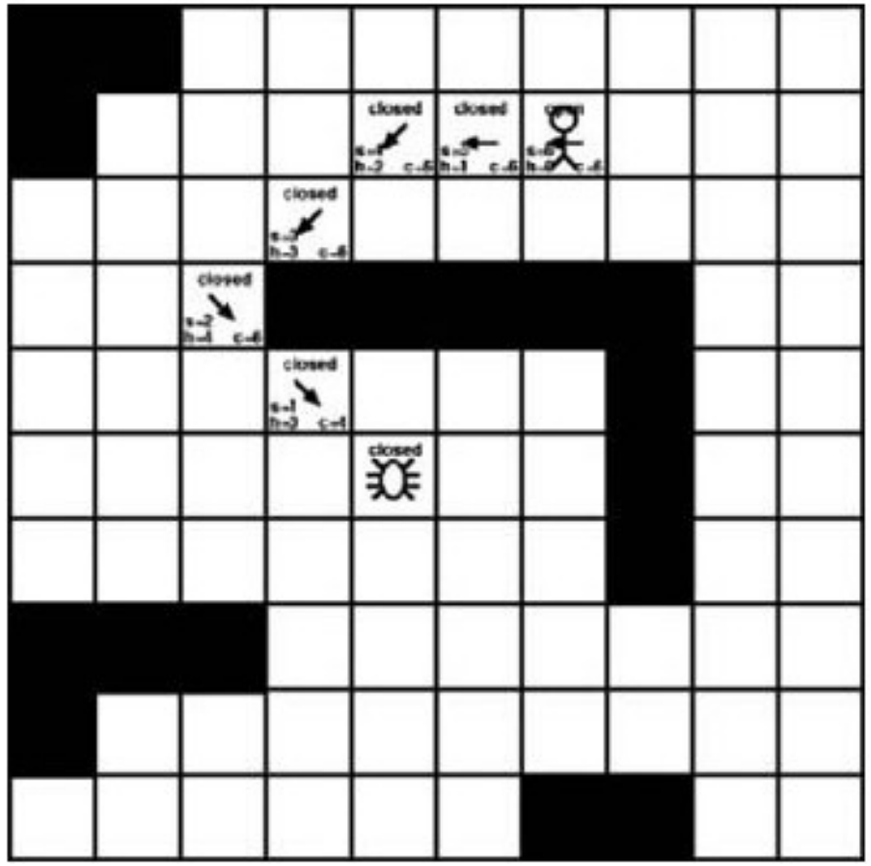
- das Zielquadrat in die „geschlossene“ Liste verschoben wird
  - erfolgreiche Ermittlung des kostengünstigsten Pfades
- kein Zielquadrat gefunden werden konnte und die offene Liste leer ist
  - es existiert kein Pfad zwischen Start- und Zielknoten:

    „Dead End“

Erfolgreiche Suche

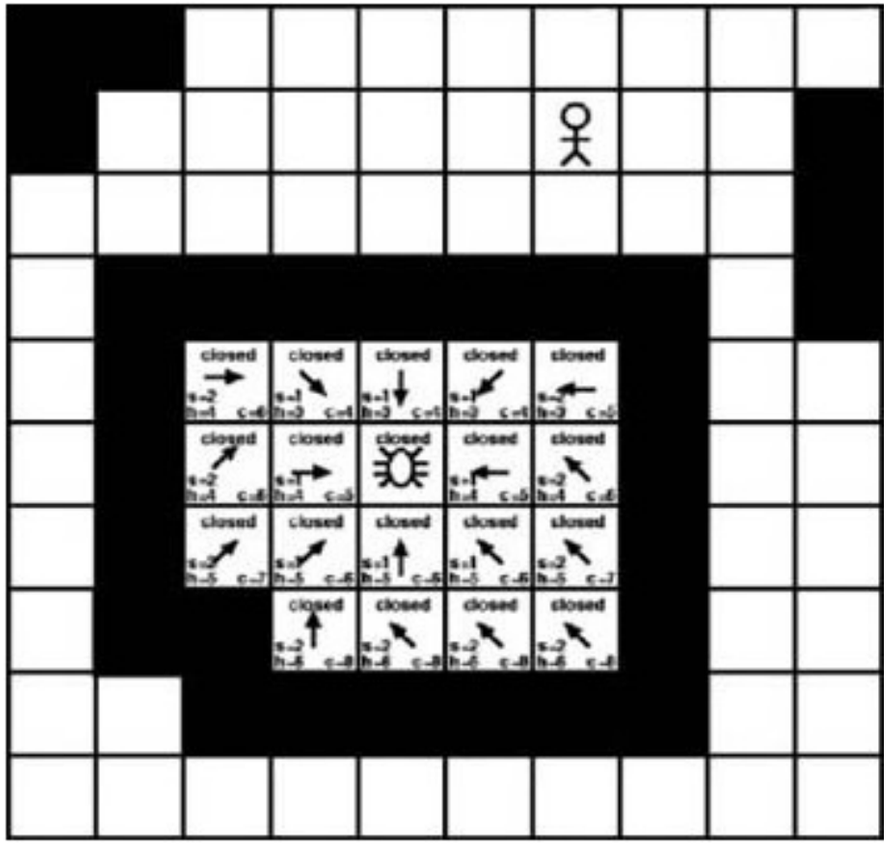


Zielknoten erreicht



Der komplette Pfad

„Dead End“



Erweiterungen

Geländekosten

- Einfluss des Geländes/Terrains auf die Pfadsuche

Influence Mapping

- "Einflusskartierung“, KI bei der Pfadfindung miteinbeziehen

Behandlung unerforschter Gebiete

- Erkundung des Geländes

Geländekosten

Kürzeste Pfad nicht immer der schnellste

- z.B. Wüste, Sumpf, Hügel usw. durchqueren.
- Bewegungskosten für solches Gelände erhöhen, z.B.:

Normales Gelände: Bewegungskosten = 10
Sumpf: Bewegungskosten = 30
Hügel: Bewegungskosten = 50

- Gelände mit unterschiedlichen Bewegungskosten

⇒ Berechnungsgleichung Ändern

Berechnungsgleichung mit Geländekosten

Ursprüngliche Gleichung:

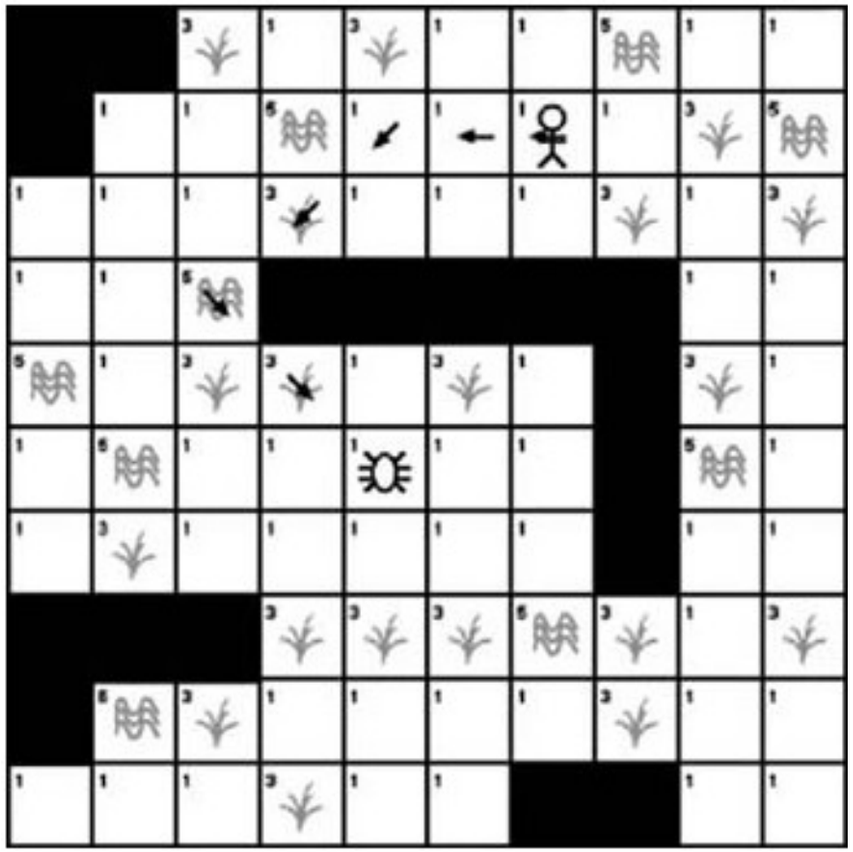
Gesamt-Wegkosten =
Wegkosten vom Startpunkt + Wegkosten zum Endpunkt

Neue Gleichung:

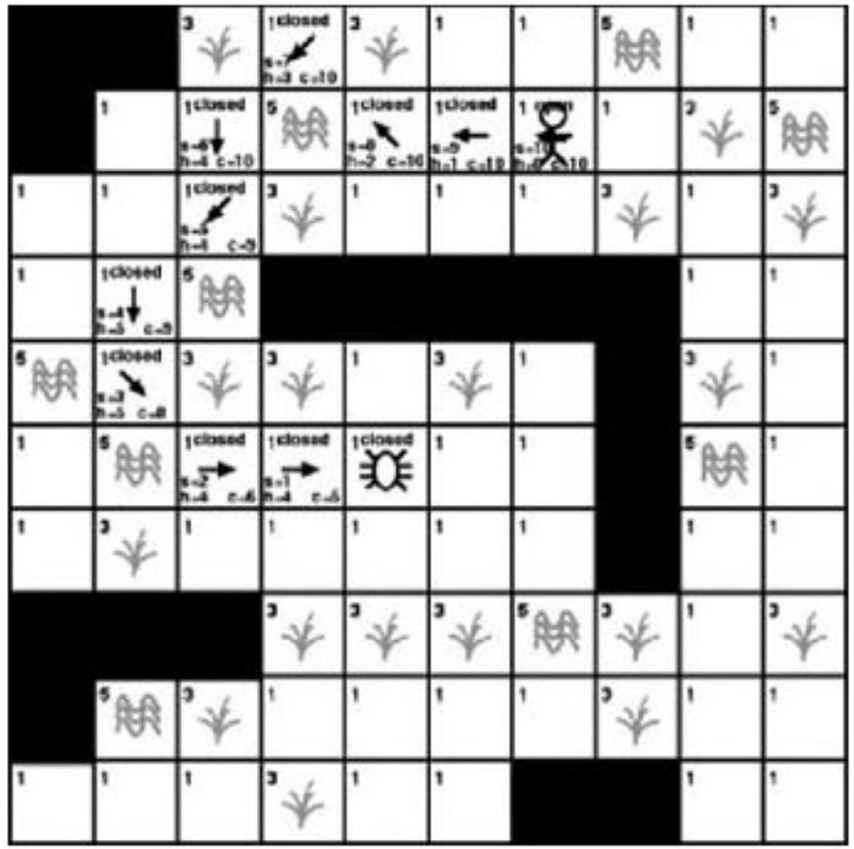
Gesamt-Wegkosten =
(Wegkosten vom Startpunkt + Geländekosten) + Wegkosten zum Endpunkt

Ergebnis: Schnellster Pfad zum Ziel wird gefunden!

Vergleich: ohne vs. mit Geländekosten



Zielpfad ohne Berücksichtigung der Geländekosten

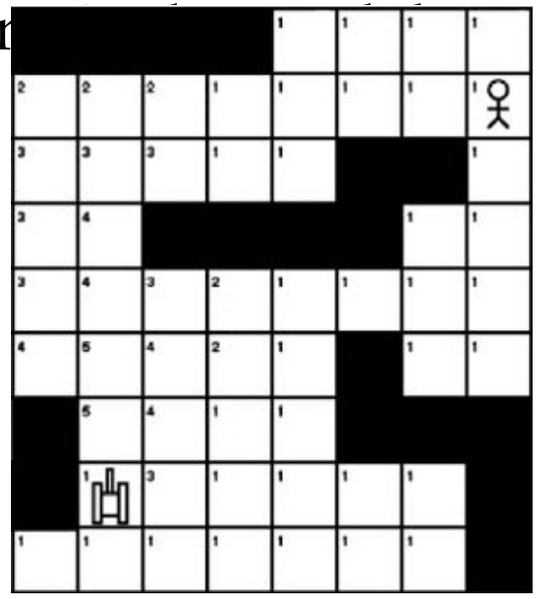


Zielpfad mit Berücksichtigung der Geländekosten

Influence Mapping 1

Punktesystem bei der Pfadfindung hinsichtlich der KI

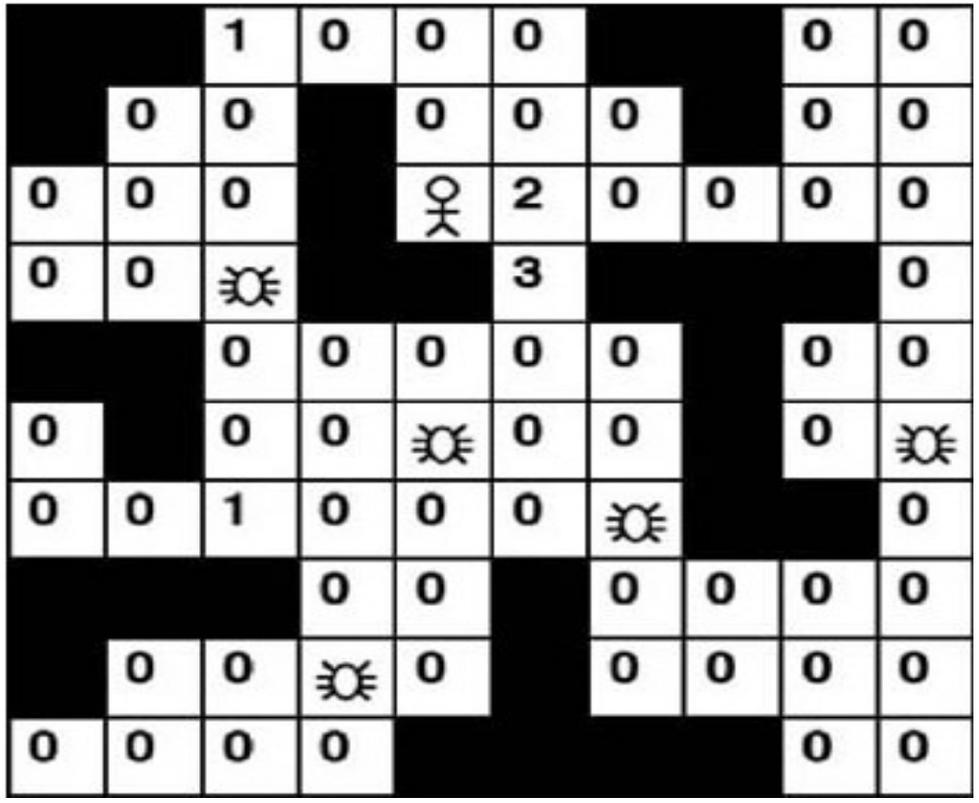
- Kosten je Knoten können variieren
- im Voraus nicht bekannt, abh. von der Spielsituation
- z. B. höhere Gefahr wenn im Blickwinkel des Feindes,
- zusätzliche Kosten pro Knoten
  Positionswechsel



Influence Mapping 2

- Merken und Festhalten von best. Ereignissen
- z. B. Spieler tötet Computergesteuerte Gegner immer an der selben Stelle.
- Erhöhung der Kosten für entspr. Knoten
- Lernen aus Fehlern und andern Pfad finden

Beispiel Influence Mapping



- Beeinflussung durch die Anzahl der Tötungen
- pro Tötung Erhöhung der Kosten um eine Einheit

Behandlung unerforschter Gebiete

Zu gutes Pathfinding kann unrealistisch wirken

- Begehbarkeit für alle Gebiete Annehmen
- sobald Gebiete erkundet, läuft Pathfinding wieder normal

Fazit

- Pathfinding-Problem mit A* Algorithmus gelöst
- A* Algorithmus findet optimalen Weg zwischen zwei Punkten

- verbraucht relativ viel CPU (bes. bei vielen Spielcharakteren)
- nicht sinnvoll, wenn einfache Umgebung/Landschaft

- ständige Weiterentwicklung, noch schneller und effizienter, Fokus auf Optimierung

Demo Remote Assault

Quellen

- David M. Bourg, Glenn Seemann: AI for Game Developers, O'Reilly, 2004; Kap. 7
- Patrick Lester: A* Pathfinding for Beginners; http://www.policyalmanac.org/games/aStarTutorial.htm
- Amit J. Patel: http://theory.stanford.edu/~amitp/GameProgramming
- http://de.wikipedia.org/wiki/A%2A-Algorithmus
- Demo A* Algorithmus: http://www.blitzcoder.com/cgi-bin/showcase/showcase_showentry.pl?id=turtle1776040620020 02208&comments=no

Vielen Dank für die Aufmerksamkeit !