[MI 2173]
Schliep, Jan

- *Karte / Graph* – Bereich, den der A* - Algorithmus verwendet, um einen Weg zwischen zwei Positionen zu finden
- *Knoten* – Repräsentieren Positionen auf der Karte und sind Datenstrukturen zum Hinterlegen der Feldinformationen
- *Distanz / Heuristik* – Ermittelt die Tauglichkeit eines Knotens
- *Kosten* – Auftretende Kosten, um von einer Position auf der Karte zu einer anderen zu gelangen

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Seminar – Multimediale Anwendungen 2

Titel: **Spiele – KI**

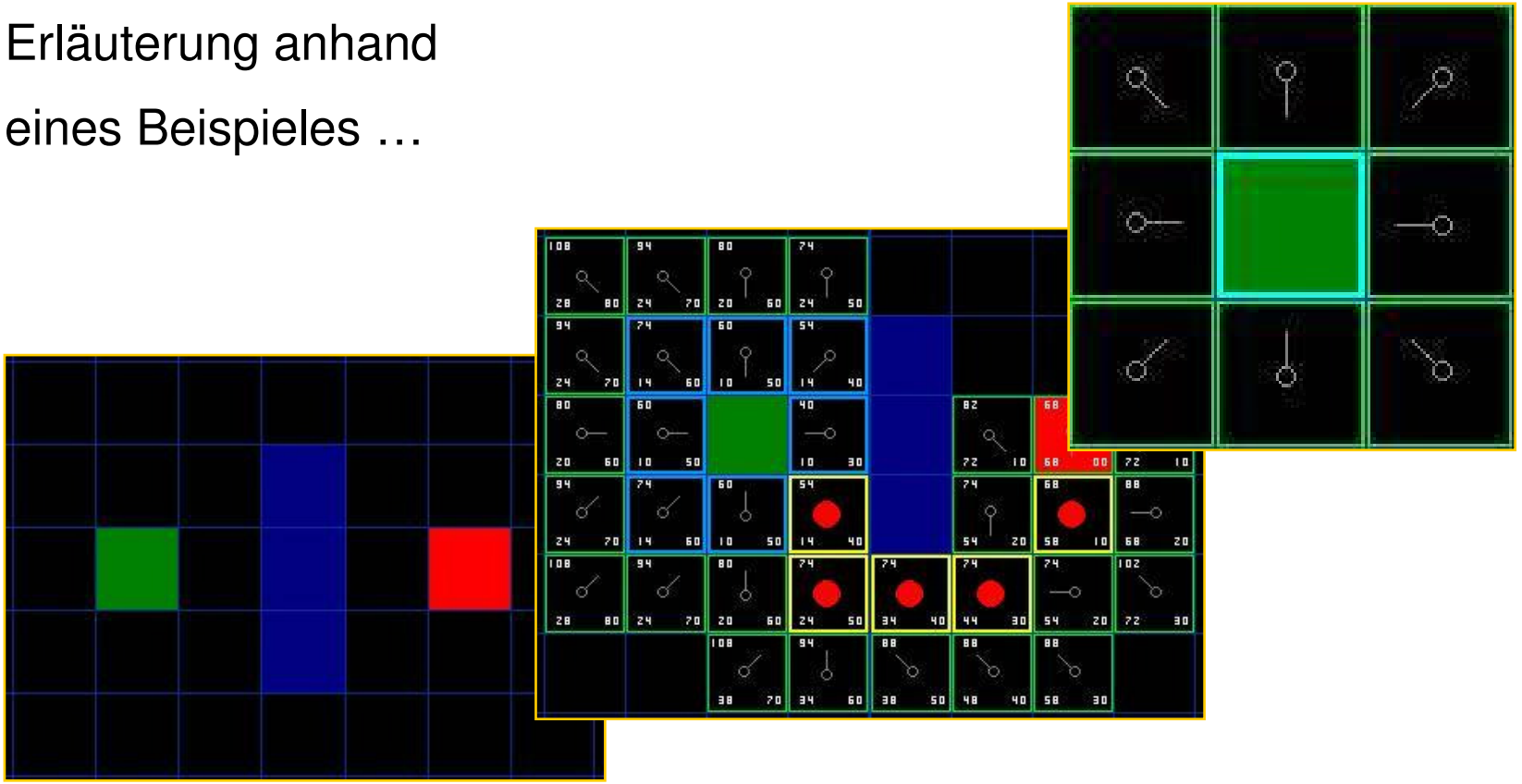
Thema 4: **Anpassung des A* - Algorithmus an reale Anwendungen**

[MI 2173]

Schliep, Jan

Erläuterung anhand

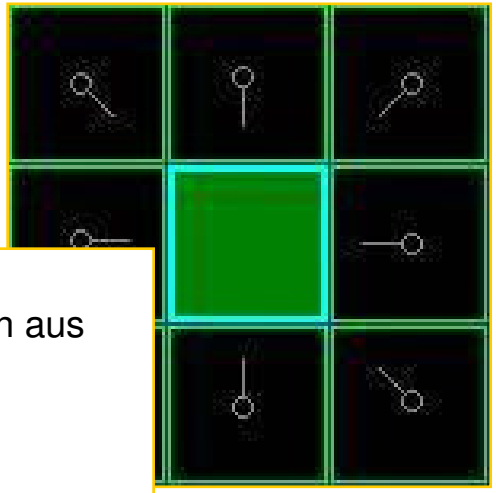
eines Beispieles …

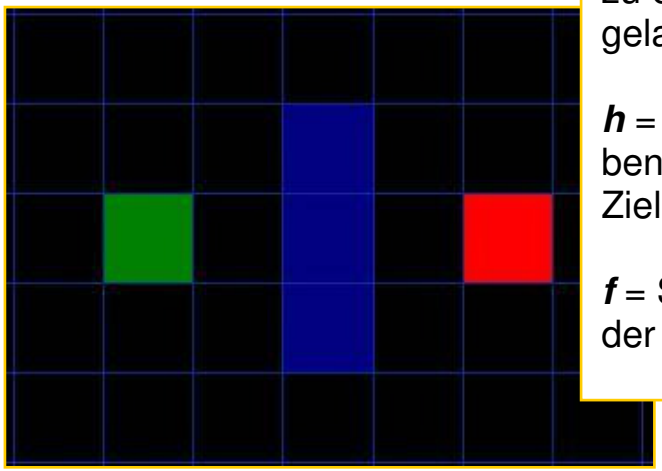


[MI 2173]
Schliep, Jan

Erläuterung anhand

eines Beispieles …





g = Bewegungskosten – werden benötigt, um von dem Startknoten aus zu einem spezifischen Knoten zu gelangen

h = Geschätzte Kosten – werden benötigt, um von einem Knoten zu dem Zielknoten zu gelangen

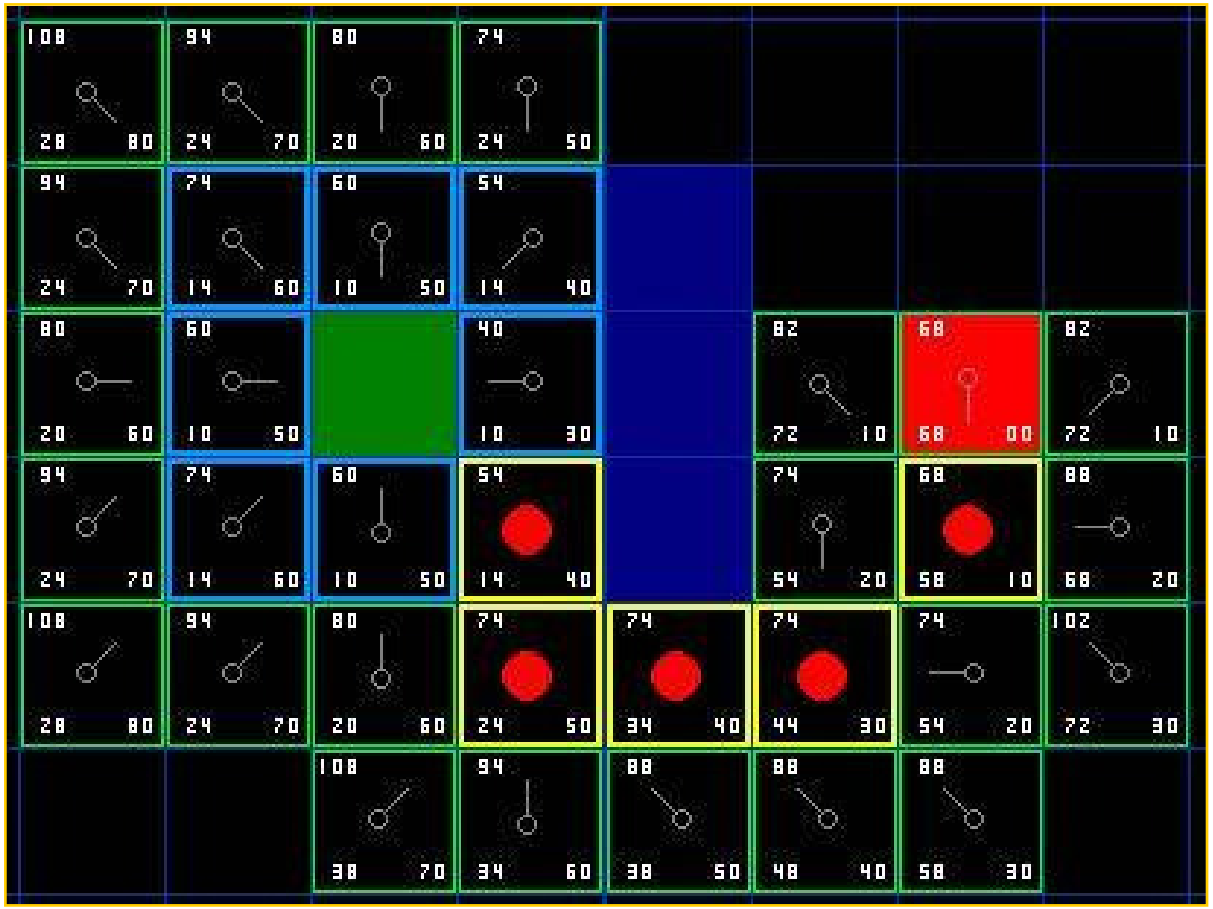
f = Summe der geschätzten Kosten und der anfallenden Bewegungskosten

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Seminar – Multimediale Anwendungen 2

Titel: **Spiele – KI**

Thema 4: **Anpassung des A* - Algorithmus an reale Anwendungen**



[MI 2173]
Schliep, Jan

[MI 2173]
Schliep, Jan

1. S = Startknoten
2. Ermittlung der g, h und f Werte für S
3. Füge S der „offenen Liste“ hinzu
4. B = erfolgversprechenster Knoten in der „offenen Liste“ (niedrigster f Wert)
    a) Wenn B der Zielknoten ist – brich ab (optimaler Pfad wurde gefunden)
    b) Wenn die „offene Liste“ leer ist – brich ab (kein Pfad gefunden)
5. C = gültiger Nachbarknoten von B
    a) Ermittlung der g, h und f Werte für C
    b) Überprüfung, ob C in der „offenen“ oder „geschlossnen Liste“ ist
        i. Wenn dem so ist – Überprüfung, ob der Weg effizienter ist (niedrigerer f Wert)
        -> Wenn dem so ist – Aktualisierung des Pfades
        ii. Ansonsten füge C der „offenen Liste“ hinzu
    c) Wiederhole Schritt 5 für alle gültigen Nachbarknoten von B
6. Wiederhole Schritt 4 und folgende Schritte

[MI 2173]
Schliep, Jan

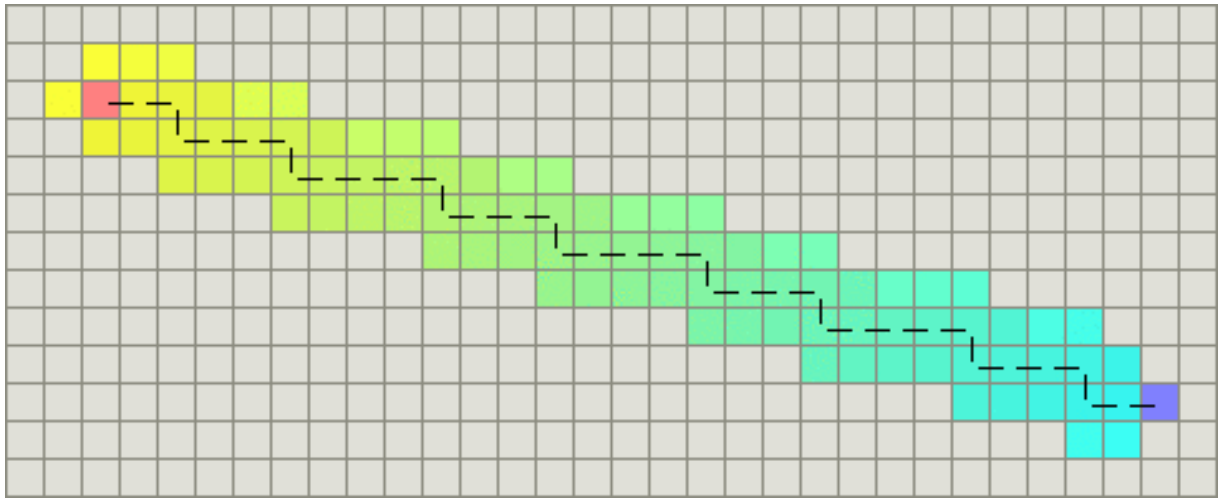
- Unterscheidung der Heuristik in zulässig und monoton

  - Zulässig – Kosten zum Ziel dürfen nicht, Kosten zwischen Knoten dürfen aber sehr wohl überschätzt werden
  - Monoton – Weder die Kosten zum Ziel, noch die Kosten zwischen den Knoten dürfen überschätzt werden

[MI 2173]
Schliep, Jan

- Manhattan – Methode

$$h = 10 * (abs (currentX – targetX) + abs (currentY – targetY))$$



[MI 2173]
Schliep, Jan

- Diagonal Shortcut

```
xDistance = abs ( currentX – targetX )
yDistance = abs ( currentY – targetY )
if xDistance > yDistance
    h = 14 * yDistance + 10 * ( xDistance – yDistance )
else
    h = 14 * xDistance + 10 * ( yDistance – xDistance )
end if
```

[MI 2173]
Schliep, Jan

- Je geringer h gegenüber g, desto mehr Knoten werden überprüft
- Ist h genau der verbleibenden Distanz, wird nur der optimale Pfad betrachtet – keine weiteren Knoten
- Ist h größer als die verbleibende Distanz, kann nicht gewährleistet werden, dass auch wirklich der optimale Pfad gefunden wird

- Eine gute Heuristik stellt einen Kompromiss zwischen dem Berechnungsaufwand für das Verfahren und der Genauigkeit dar!

[MI 2173]
Schliep, Jan

Exkurs [Empire Earth] :

- Genre – Echtzeit Strategiespiel
- Ziel – Eine ausgewählte Zivilisation erfolgreich durch die verschiedenen Epochen der Menschheitsgeschichte führen



[MI 2173]
Schliep, Jan

- A* - Algorithmus kann nicht nur zur Ermittlung des optimalen Pfades bezüglich der Wegfindung genutzt werden

  - Terrainanalyse
  - Choke – Point Detection
  - Planung der KI Militärrouten
  - Planung und Bewegung von Tieren
  - Erschaffung und Bewegung des Wetters
  - Bauen von Mauern und Verteidigungsanlagen durch die KI

- Um nun eine generische A* Engine, bzw. Machine zu erhalten, sind folgende Faktoren notwendig

  - Der Speicher
  - Das Ziel
  - Die Karte

- Manövrieren hunderte Einheiten über die Karte des Spiels, wird das System extrem belastet
- Pfadfindung soll aber trotzdem komplex angewendet werden, ohne das System dabei in die Knie zu zwingen
- Soll eine Einheit einen weiten Weg auf der Karte zurücklegen, muss dieser zunächst ermittelt werden
- Erwartungshaltung des Spielers ist aber: „Anklicken und loslaufen“
- Problematik des Einfrierens von Einheiten und der Framerate
  - ➢ Lösung: Dreistufiges Aufteilung des Pfades über die Zeit

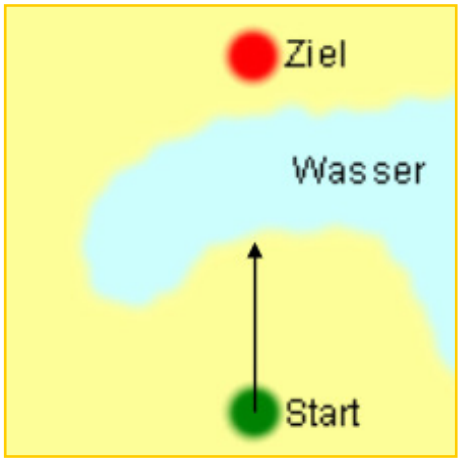
- Quickpath

➢ Highspeed Pfadfinder für kurze Distanzen

➢ Nur bestimmte Anzahl an Revolutions für die Wegfindung

➢ Es wird der Punkt ausgewählt, der am nächsten am Ziel liegt

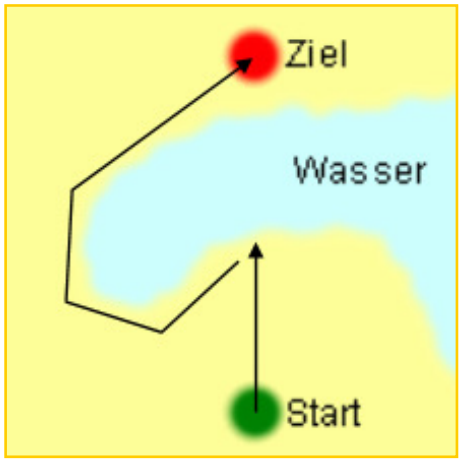
➢ Die Einheit bewegt sich auf diesen ermittelten Punkt als Zielpunkt zu



- Fullpath

➢ Wird im Hintergrund durch den A* - Algorithmus ermittelt

➢ Startknoten ist der zuvor ermittelte Endknoten des Quickpath



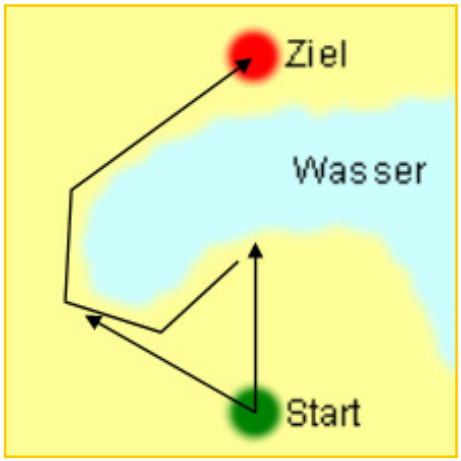
- Splice Path

➢ Anpassung an realistischeres Verhalten

➢ Highspeed Pfadfinder für kurze Distanzen

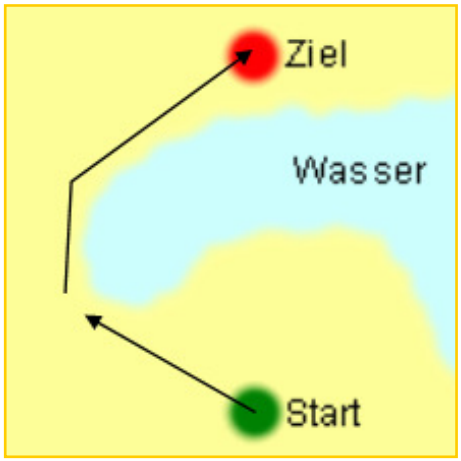
➢ Startknoten ist die aktuelle Position der Einheit auf der Karte

➢ Punkt als Ziel für den Splice Path ergibt sich aus Erfahrungswerten



- Entfernen der Extra - Waypoints

➢ Abschließend werden die zusätzlichen Waypoints entfernt



- Größtes Problem der Performance ist der Speicherzugriff

  - Erstellung einer Speicherklasse mit einer eigenständigen Listenstruktur anstelle der Standard Template Library (C++)
  - Verwaltung eines Memory – Pools durch eine eigene Klasse
  - Optimierung des Cache durch 1 – Byte Flag Arrays
  - Hash – Tabellen für die Listenverwaltung
  - Optimierung der „offenen Liste“

- Reduzierung der iterativen Tiefe des A* - Algorithmus
- Blockierte Knoten werden umgehend geschlossen
- Knoten mit zu hohen Kosten werden geschlossen
- Ermittlung eines Teilweges anstelle des vollständigen Pfades
- Cachen von fehlgeschlagenen Wegfindungen
- Begrenzung der Zeit, die in den A* - Algorithmus investiert wird
- Verwendung von Waypoints, um lange Wege zu generieren

- Reduzierung der iterativen Tiefe des A* - Algorithmus
- Blockierte Knoten werden umgehend geschlossen
- Knoten mit zu hohen Kosten werden geschlossen
- Ermittlung eines Teilweges anstelle des vollständigen Pfades
- Cachen von fehlgeschlagenen Wegfindungen
- Begrenzung der Zeit, die in den A* - Algorithmus investiert wird
- Verwendung von Waypoints, um lange Wege zu generieren

- Reduzierung der iterativen Tiefe des A* - Algorithmus
- Blockierte Knoten werden umgehend geschlossen
- Knoten mit zu hohen Kosten werden geschlossen
- Ermittlung eines Teilweges anstelle des vollständigen Pfades
- Cachen von fehlgeschlagenen Wegfindungen
- Begrenzung der Zeit, die in den A* - Algorithmus investiert wird
- Verwendung von Waypoints, um lange Wege zu generieren

- Reduzierung der iterativen Tiefe des A* - Algorithmus
- Blockierte Knoten werden umgehend geschlossen
- Knoten mit zu hohen Kosten werden geschlossen
- Ermittlung eines Teilweges anstelle des vollständigen Pfades
- Cachen von fehlgeschlagenen Wegfindungen
- Begrenzung der Zeit, die in den A* - Algorithmus investiert wird
- Verwendung von Waypoints, um lange Wege zu generieren

- Reduzierung der iterativen Tiefe des A* - Algorithmus
- Blockierte Knoten werden umgehend geschlossen
- Knoten mit zu hohen Kosten werden geschlossen
- Ermittlung eines Teilweges anstelle des vollständigen Pfades
- Cachen von fehlgeschlagenen Wegfindungen
- Begrenzung der Zeit, die in den A* - Algorithmus investiert wird
- Verwendung von Waypoints, um lange Wege zu generieren

- Reduzierung der iterativen Tiefe des A* - Algorithmus
- Blockierte Knoten werden umgehend geschlossen
- Knoten mit zu hohen Kosten werden geschlossen
- Ermittlung eines Teilweges anstelle des vollständigen Pfades
- Cachen von fehlgeschlagenen Wegfindungen
- Begrenzung der Zeit, die in den A* - Algorithmus investiert wird
- Verwendung von Waypoints, um lange Wege zu generieren

- Reduzierung der iterativen Tiefe des A* - Algorithmus
- Blockierte Knoten werden umgehend geschlossen
- Knoten mit zu hohen Kosten werden geschlossen
- Ermittlung eines Teilweges anstelle des vollständigen Pfades
- Cachen von fehlgeschlagenen Wegfindungen
- Begrenzung der Zeit, die in den A* - Algorithmus investiert wird
- Verwendung von Waypoints, um lange Wege zu generieren