Basic Pathfinding and Waypoints

Chapter 6, AI for Game Developers, Bourg&Seeman

von Wladimir Awerbuch

Gesamtübersicht

1. Definition von „Pathfinding“
2. Einleitung
3. Basic Pathfinding
4. Breadcrumb Pathfinding
5. Path Following
6. Wall Tracing
7. Waypoint Navigation
8. Zusammenfassung und Ausblick
9. Eigene Erfahrungen

1 Definitionen von Pathfinding

- The process of finding a path between an origin and destination, which usually involves determining a least-cost path. (GIS-Glossary)
- **Pathfinding** is a term used mostly by computer applications to plot the best route from point A to point B. (Wikipedia)

2 Einleitung

- Es gibt viele Anwendungen für Pathfinding Algorithmen

2 Einleitung

- Aber:

- die Lösung hängt oft von verschiedensten Faktoren ab
  - Ziel beweglich oder fest?
  - Hindernisse vorhanden? Sind Sie beweglich?
- Die kürzeste Lösung ist nicht immer die beste
  - Terrain muss beachtet werden
  - Positionierung der Gegner muss beachtet werden
- Ziel muss nicht immer vorhanden sein
  - erforschen
  - auf der Strecke bleiben

=>1. Es gibt keine universelle Lösung
2. Es gibt meist mehr als einen möglichen Lösungsansatz

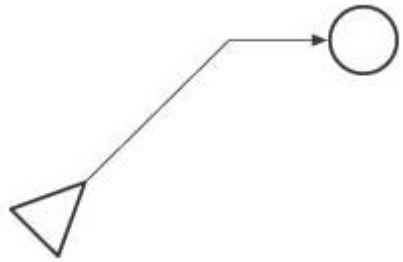
3 Basic Pathfinding

- Die unterste Ebene des „Pathfindings“
- Prozess des Bewegens der Spielcharakterposition von seiner Startposition zum Wunschziel
- Algorithmus:

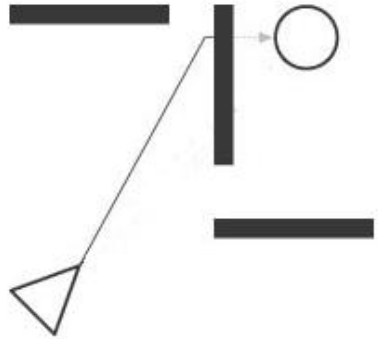
```
if(positionX > destinationX)        positionX--;
else if(positionX < destinationX)   positionX++;
if(positionY > destinationY)        positionY--;
else if(positionY < destinationY)   positionY++;
```

3 Probleme des BPA

- Der Pfad sieht sehr unnatürlich aus

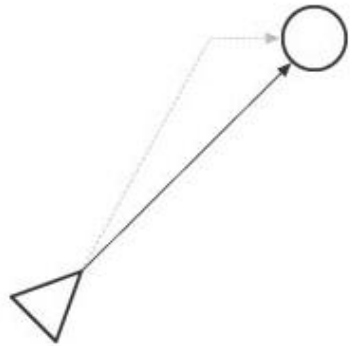


- Hindernisse

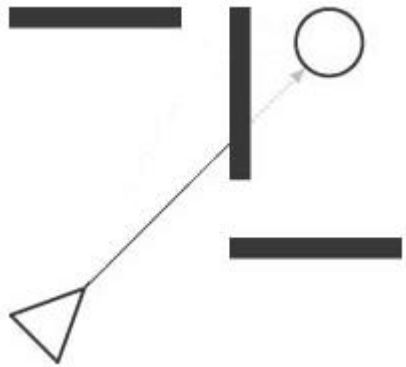


3 Line of Sight Path Movement

- Behebt das Problem mit der „Unnatürlichkeit“ des Pfades.



- Aber nicht das Problem mit Hindernissen



3 Random Movement Obstacle Avoidance

- Erste Lösung um Hindernisse zu umgehen
- Einfach und Effektiv
- Funktioniert gut bei relativ wenigen Hindernissen

- Algorithmus:

if Player In Line of Sight

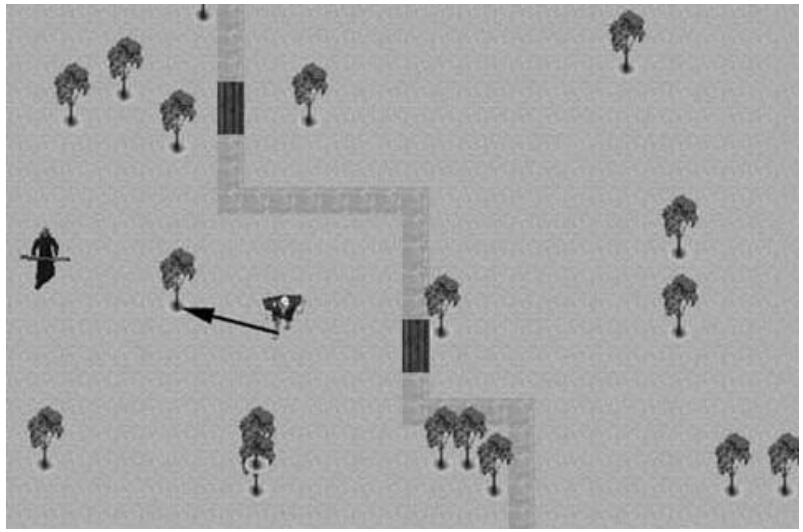
{ Follow Straight Path to Player }

else

{ Move in Random Direction }

3.3 Random Movement Obstacle Avoidance

Beispiel:



3 Tracing

- Alternative um Hindernisse zu umgehen
- Effektiv um große Hindernisse zu umgehen

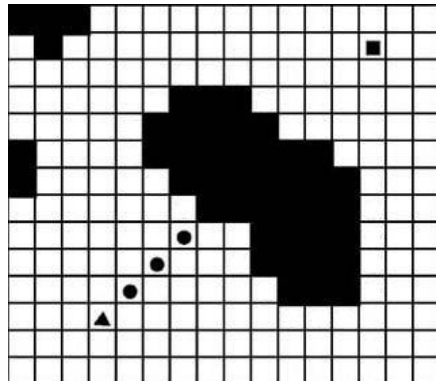
- Idee:
  - 2 States die sich gegenseitig aktivieren können:

1. Pathfinding State
2. Tracing State

3 Tracing

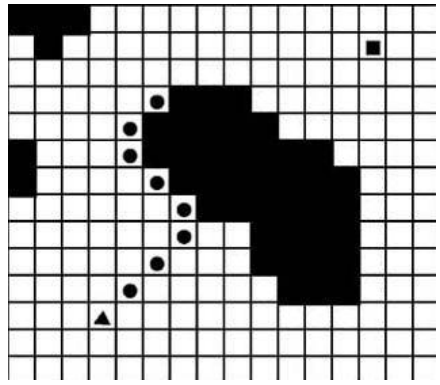
- Pathfinding State:

  Setze deinen weg fort, bis du auf ein Hindernis triffst. Wenn du auf ein Hindernis triffst, schalte ins Tracing State



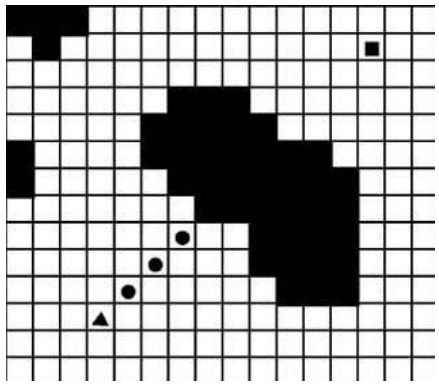
- Tracing State:

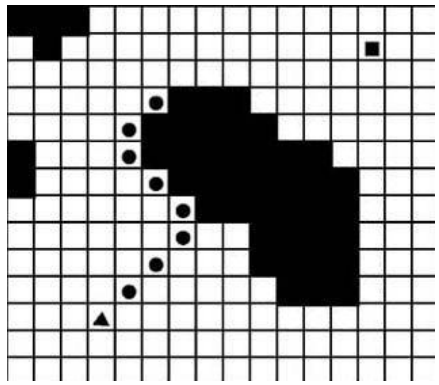
  Gehe an der Mauer des Hindernisses entlang und versuche es zu umgehen

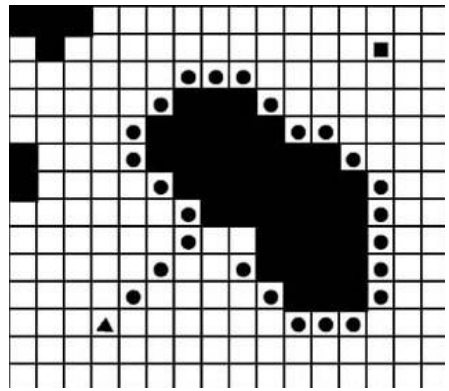


3 Problem im Tracing

- Wann soll mann mit Tracing aufhören und in „pathfinding state“ zurückschalten?







Möglichen Lösungen:

1. Tracing with a calculated line
2. Tracing with line of sight

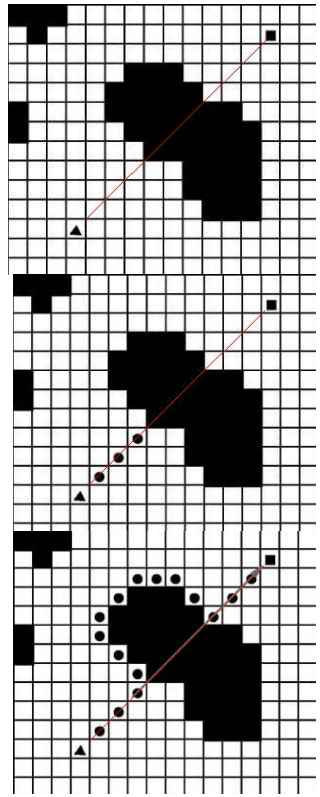
3 Tracing with a calculated line

- Idee:

1. Kalkuliere eine Luftlinie vom Start zum Zielpunkt

2. Gehe in den „Pathfinding State“ bis zum Hindernis, danach schalte in den „Tracing State“

3. Benutze Tracing bis die kalkulierte Luftlinie berührt wird, schalte an dieser Stelle in den „Pathfinding State“ zurück.



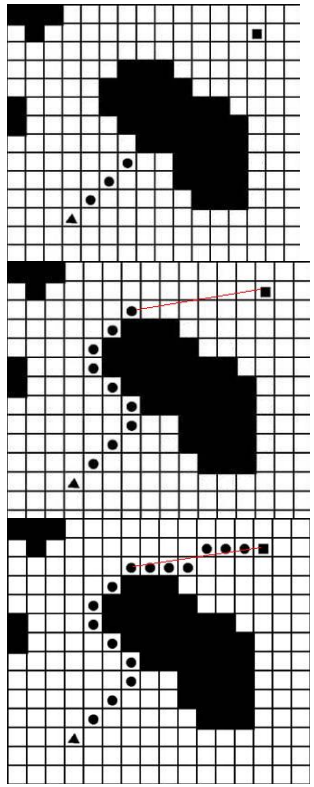
3 Tracing with line of sight

- Idee:

1. Pathfinding State bis zum Hindernis

2. Tracing State bis eine Sichtlinie zwischen Objekt und Zielpunkt besteht.

3. Führe den Bresenham-Algorithmus durch und versuche diese Linie zum Ziel möglichst genau zu imitieren.



4 Breadcrumb Pathfinding-Idee

- Spieler erstellt unwissentlich einen Weg für Computerkontrollierten Charakter

- Idee:
- Spieler hinterläßt nach jedem Schritt einen unsichtbaren Marker (Breadcrumb) auf der Karte
- Kommt ein Computercharakter in Kontakt mit dem Breadcrumb folgt er diesem Weg...
- ...solange bis er den menschlichen Spieler erreicht hat

\+ Komplexität des Pfades irrelevant
\+ Anzahl der Hindernisse irrelevant
-> Man folgt einfach (meistens!) dem Weg des Spielers

- Umsetzung in Spielen: u.A. Metal Gear Solid

4 Breadcrumb Pathfinding-Einleitung

- Ebenso effektiv um Gruppen von Computercharakteren zu bewegen (Follow The Leader Prinzip)

- Idee:

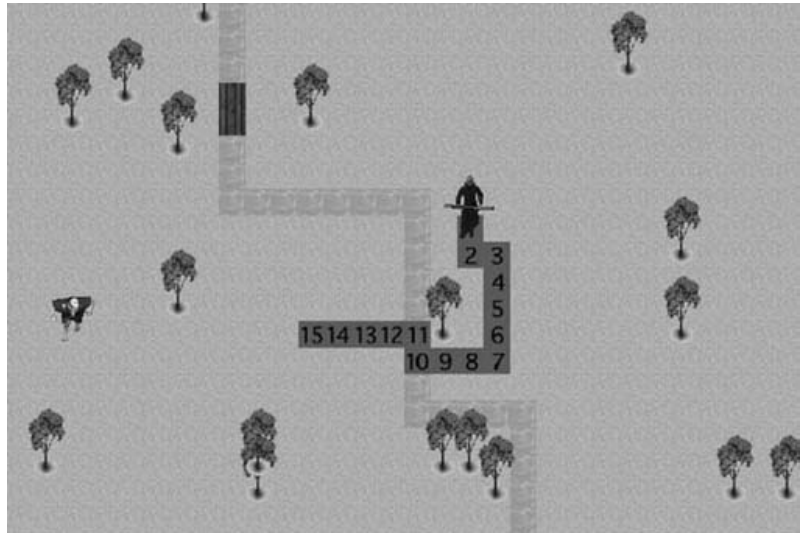
- Eine erste Einheit (Leader) macht den Weg vor und legt Breadcrumbs auf der Karte
- Die restlichen Gruppenmitglieder folgen diesen Weg anhand von Breadcrumbs

- Vorteile/Nachteile:

+ Effizient
- Der ganze Weg von einer Einheit abhängig

4 Breadcrumb Pathfinding-Einleitung

- Beispiel:
  - Breadcrumb: Integerwert
  - Max. Anzahl Aufzeichnungsschritte:15



- Realität:
  - Keine Breadcrumb-Spur sichtbar
  - Es gibt deutlich mehr Breadcrumbs

4 Breadcrumb Pathfinding-Umsetzung

- In der Class sind (unter anderem) folgende Angaben wichtig:

- Vordefinierte Anzahl von Schritten
- Reihe und Spalte der aktuellen Koordinaten
- 2 Arrays (Spalte, Zeile), in dem die Koordinaten der Breadcrumbs gespeichert werden

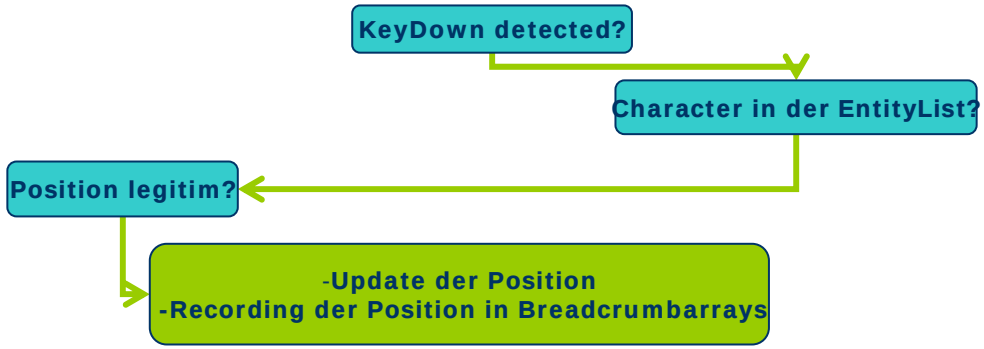
- Umsetzungsschritte:

1. Initialisierung (erfolgt durch das Setzen auf -1 der Breadcrumb-Koordinatenarrays)
2. Speichern der Spielerpositionen und Breadcrumbs legen
3. Breadcrumbs finden
4. Folgen einer Breadcrumbspur

4 Speichern der Spielerposition

- Keydown Funktion
  - prüft ob einer der Richtungstasten gedrückt wurde
  - ändert die Position wenn ein „KeyDown-Event“ auftritt

- Prüfverfahren zur Speicherung der Schritte:



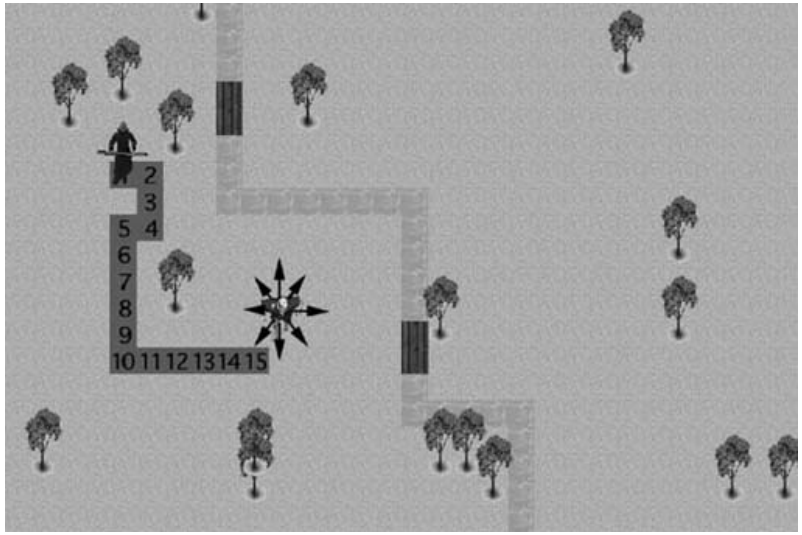
4 BreadcrumbArrays

- Addiert die aktuelle Spielerposition in die 2 Arrays (Spalte, Zeile) hinzu
- Jedes Mal bei neuer Position, Arrayshift nach rechts
- Neuste Position ganz links gespeichert
- Bei vollem Array wird die älteste Position die über den Arrayrand rausgeshiftet und gelöscht
- Anzahl der gespeicherten Spielerpositionen abhängig von Anfangsparametern

-> Beinhaltet eine Liste der letzten Spielerpositionen

4 Finden und Folgen einer Breadcrumb Trail

1. Zufällige Bewegung des Computers in alle 8 möglichen Richtungen auf der Karte (andere Verfahren möglich)
2. Prüfe ob die umliegenden Rasterkoordinaten (1 Feld) in unseren BreadcrumbArrays vorliegen.
   - Wenn gefunden, folge der Spur
   - Wenn nicht gefunden, bewege dich weiter zufällig



4 Umsetzung „Finden“ und „Folgen“

Umsetzung von „Finden:“

- Prozedur „foundcrumb“ zeigt die Anzahl der gefundenen Breadcrumbs in den umliegenden Feldern an.
- Foundcrumb=-1, keine Breadcrumbs in den umliegenden Rastern
- Foundcrumb>=0, Computer Position->Position die im BreadcrumbArray
- Mehrere Breadcrumbs auf den umliegenden Rastern->Benutze den „frischesten“ Breadcrumb.

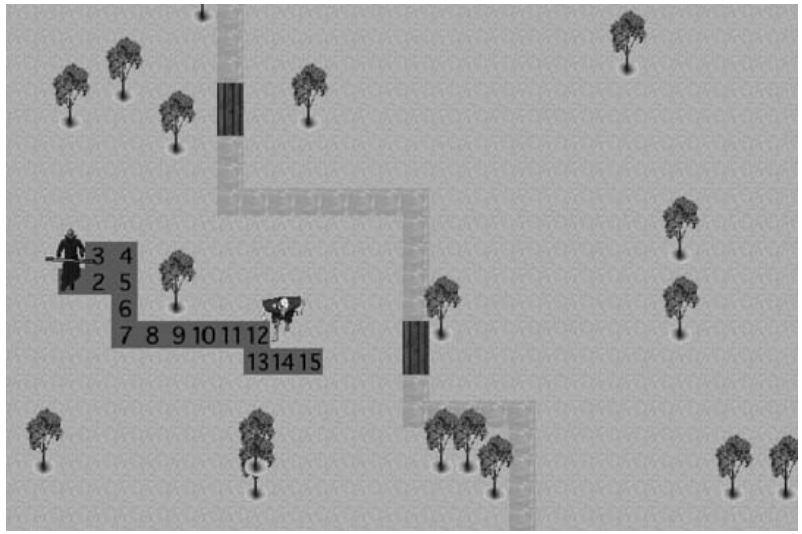
Umsetzung von „Folgen“

- Prozedur wird nach jedem Schritt ausgeführt→Computer folgt der Breadcrumbspur immer weiter

4 Probleme in der Realität

Sehr wahrscheinliche Scenarios:

- Computer findet einen Breadcrumb in der Mitte der Spur
- Spielerweg überlappt sich



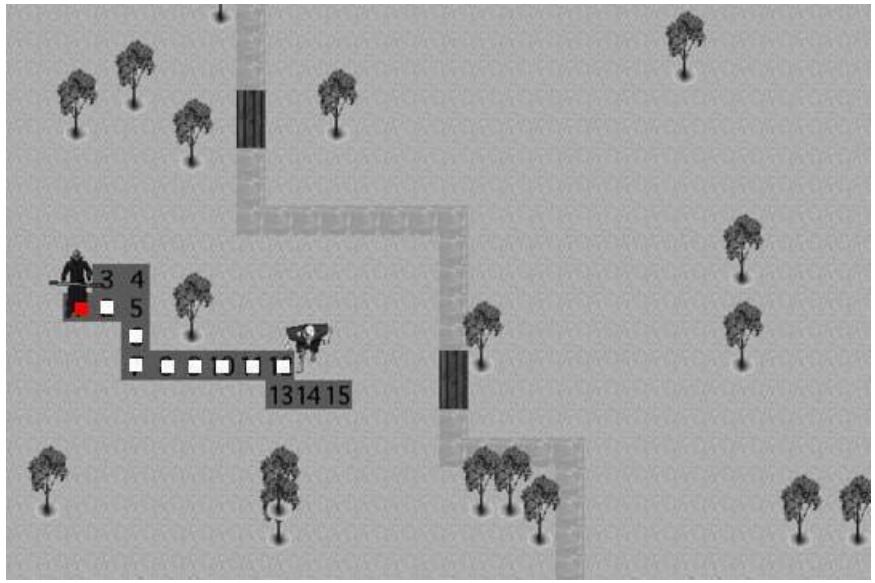
4 Lösung durch Vorteile der Abspeicherungsmethode

1. Vorteil: Spieler sucht immer nach den frischesten Breadcrumbs

-> Start bei 12

2. Vorteil: Hintereinanderausführung der Prozedur (statt einfacher Arrayabarbeitung)

->Abkürzung bei 6, spart 3 Züge



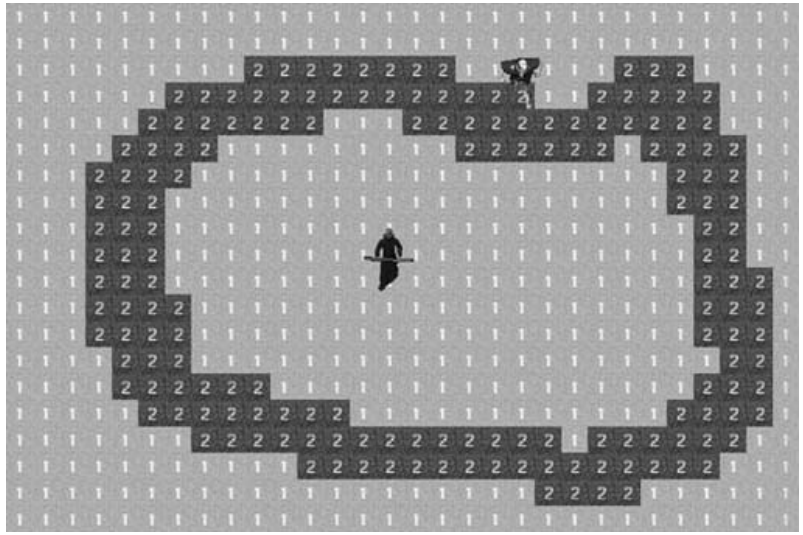
5 Path Following - ein Beispiel

- Pathfinding ist nicht nur Wegefinden vom Start zum Ziel
- Beispiel Autorennen:

- Das Auto muss vom Computer möglichst auf der Straße gehalten werden
- Auch Ideallinie sollte gefahren werden.

5 Realisierung einer Straße

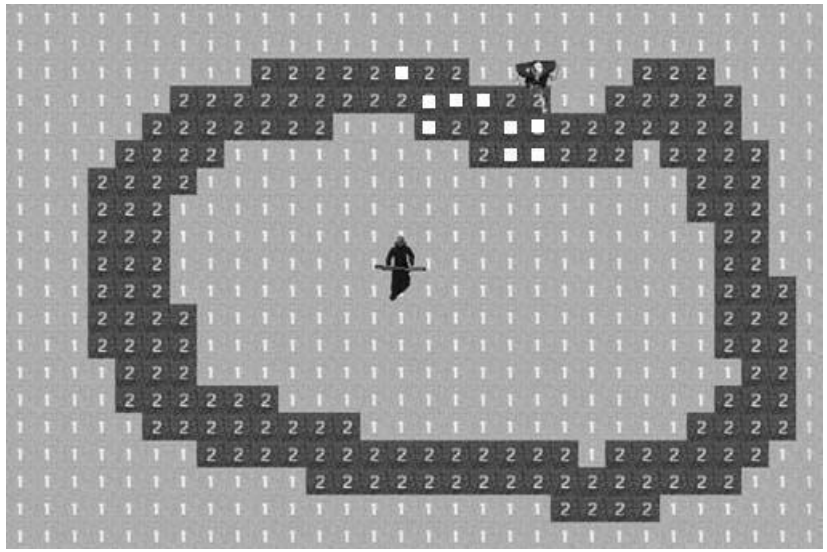
- Eine Straße wird folgendermaßen realisiert:
- „2“ Straße
- „1“ Out of Bounds



- Ziel: Realistische (Natürliche) Bewegung des Computers auf der Straße

5 Falsche Ansätze

- 1. Zufällige Bewegung:



-> Unnatürlich

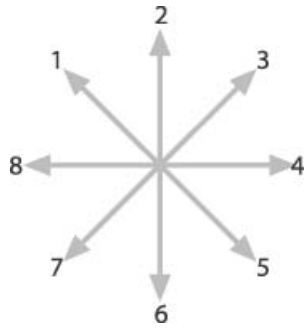
- 2. Betrachte Adjazente Kacheln nach einer 2->gehe dahin

- Straßenbreite > 1 Kachel

-> Probleme bei Entscheidungswahl

5 Path Following Ansatz

- Idee:
- Analysiere deine adjazenten Kacheln
- Entscheide dich dann für den besten Zug
- Es gibt 8 adjazente Kacheln und somit 8 Richtungen zu gehen



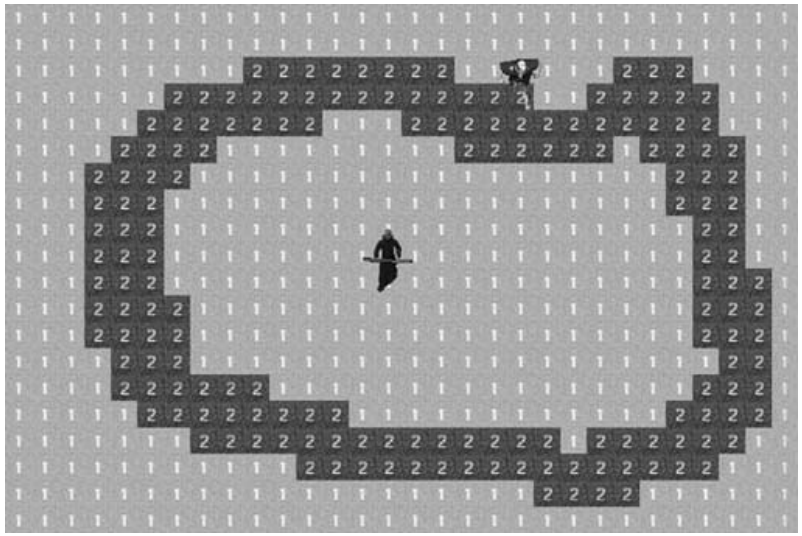
- Vorgehensweise:

1. Eliminieren von Richtungen, die nicht Teil der Straße sind
2. Gewichten von restlichen Richtungen

5 Eliminieren von „falschen“ Richtungen

- Umsetzung:

- Wir führen eine Analyse des Terrains durch:
  - -> Wir prüfen die Terrainnummer
  - -> „1“ fliegt raus, „2“ wird weiter betrachtet

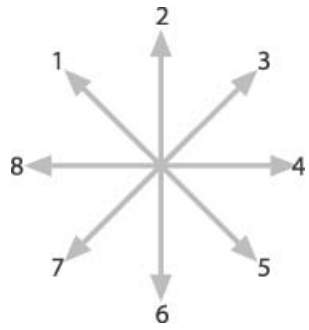


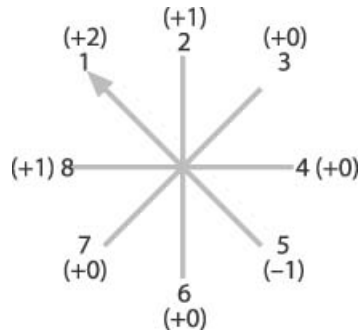
5 Gewichtung der Richtungen

- Jetzt (nach Eliminieren) gewichten wir die restlichen Richtungen
- Idee:

- Computer soll seine Richtung möglichst nicht ändern
- Änderungen möglichst marginal
- Computer soll möglichst nicht zurück gehen

-> Richtung „1“ folgende Gewichtung:

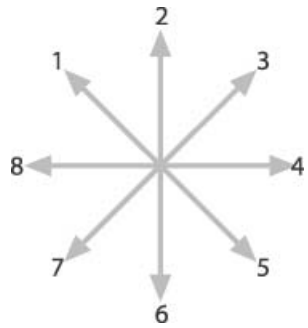




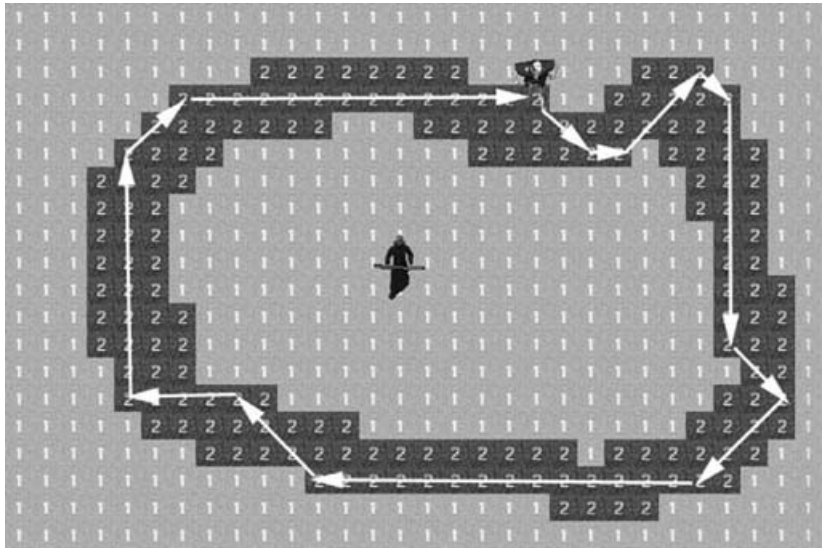
5 Weitere Schritte

- Auswahl der Richtung:
- Schleife geht alle Positionen durch
- Das größtmögliche Gewicht wird ausgewählt

- Update der Position:
- Position entsprechend der Auswahl aktualisiert
- z.B. Richtung „7“ Charakter.Reihe++, Charakter.Spalte--



5 Das Ergebnis



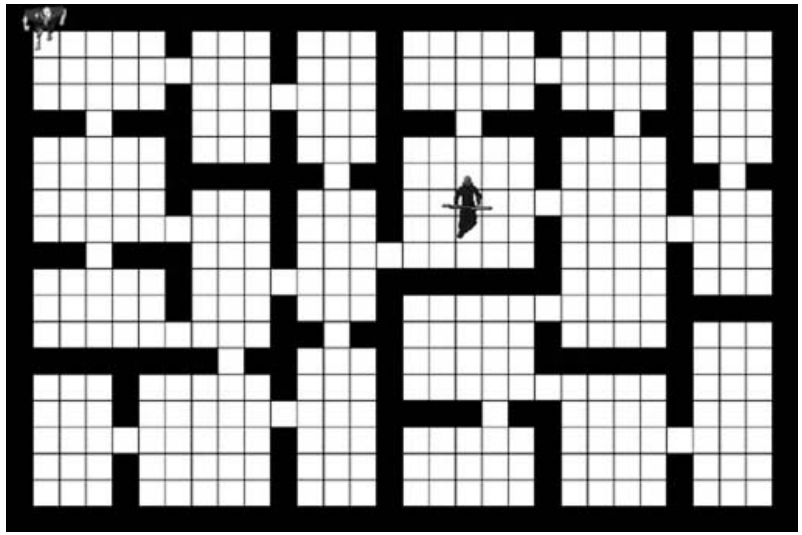
- Mögliche Verbesserung:

- Betrachte die Kacheln eine Stufe weiter

-> Führt zu einem noch natürlicherem Weg

6 Wall Tracing

- Kein direktes Ziel
- Ausgangssituation:
  - Haus mit vielen kleinen Zimmern (oder z.B. Labyrinth)
  - Wir wollen die Zimmer „erforschen“...
  - ...und dies in angemessener Zeit



6 Ansätze

- Random Movement

\+ unberechenbarer Computergegner

\- Langes Verharren in einem Zimmer

Besserer Ansatz:

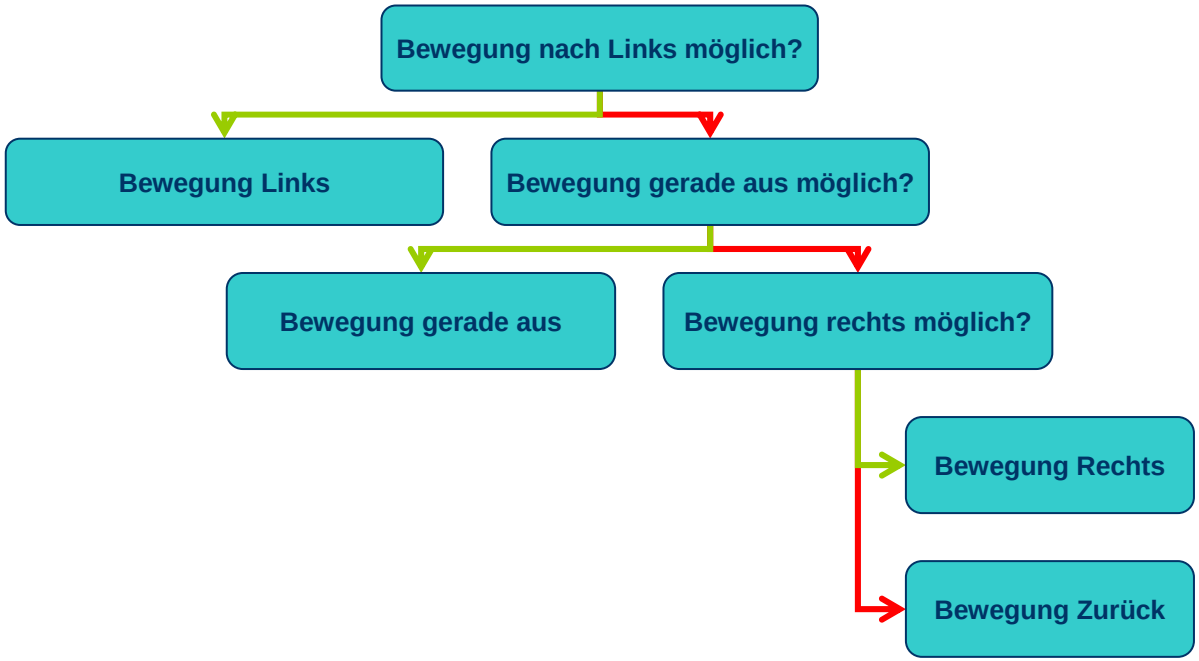
- Left-Handed Approach

\- Bewege dich möglichst links

\- Links aus der Computerperspektive

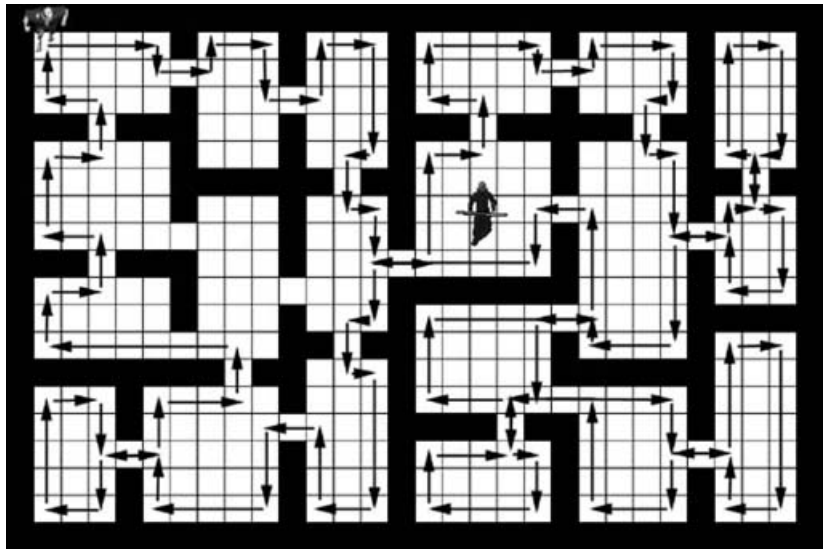
-> Systematisches Erforschen der Umgebung

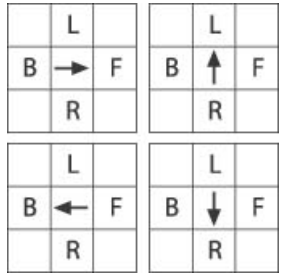
6 Left-Handed Movement Funktionsweise



6 Beispiel Left-Handed Movement

- Wir betrachten unsere Demo:



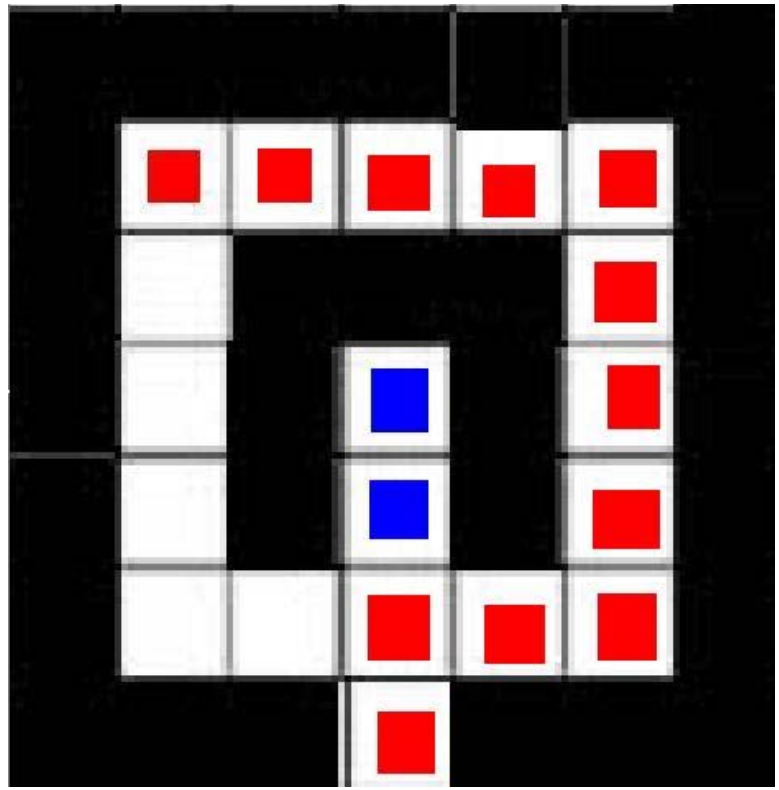


- Lösung mit Left Handed Movement

6 Vor- und Nachteile von LHM

- Vorteile
+ Einfach
+ Effektiv

- Nachteile:
- Funktioniert nicht immer
- Computer ist berechenbarer



7 Waypoint Navigation

- Pathfinding kann sehr Zeit- und Ressourcenintensiv sein
- Idee um das Problem zu reduzieren:

- Kalkuliere die Pfade möglichst vor

-> Waypoint Navigation

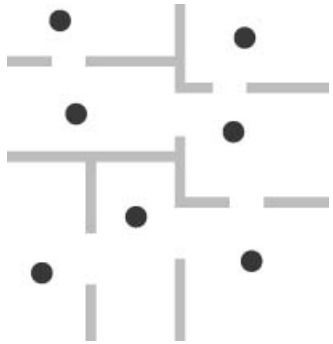
- Idee Waypoint Navigation:

1. Plaziere die Knoten in der Umgebung
2. Labeling und Verbinden der Pfade zwischen Knoten
3. Benutze vorkalkulierte Pfade oder einfache Pathfindingmethoden um sich zwischen diesen Knoten Richtung Ziel zu bewegen.

7 Plazieren der Knoten

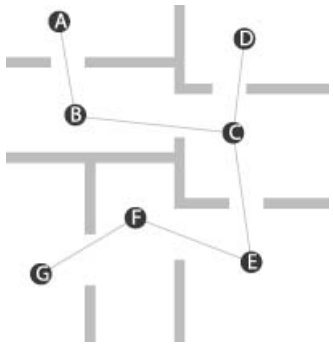
- Knotenplazierung erfolgt nicht zufällig
- Die Knoten sollten möglichst gut plaziert werden

- Möglichst gute Erreichbarkeit zwischen (mindestens) je 2 Knoten
- Gute Option ist: Line of Sight zwischen (mindestens) je 2 Knoten



7 Labeling und Verbinden

- Die Nachbarknoten (mit Line of Sight) sollten nebeneinander gelabelt werden
- Danach sollten die Knoten möglichst mit ihrem Nachbar verbunden werden



7 Kalkulieren der Pfade (im Groben)

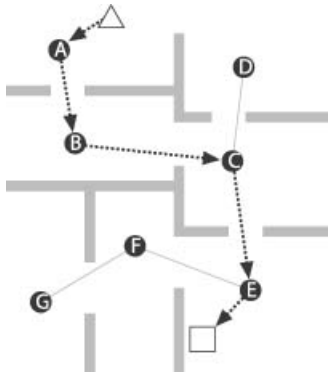
- Computer kalkuliert erst den nächstgelegenen Waypoint für Startpunkt
- Dafür betrachtet er seine Location und seine Line of Sight
- Die gleiche Kalkulation führt der Computer für das Ziel durch

-> Danach folgt der Computer den Line of Sight Wegepunkten vom Start- zum Zielpunkt

Start:A

Ziel:E

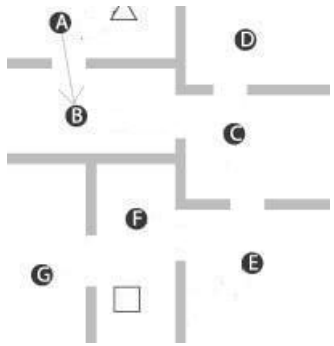
Weg: A->B->C->E



Aber: Woher weißt der Computer welche Knoten er auf dem Weg nehmen soll?

7 Tabelle der Richtungen

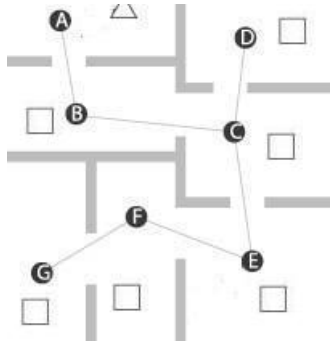
- Idee: Einfache Tabelle der Richtungen
  - Wir schauen für jeden Knoten uns nur den ersten Pfad vom Startpunkt Richtung Zielpunkt an
  - Jeder Knoten wird als Start- und Zielknoten betrachtet
- Beispiel:
  - Start A->Ziel F
  - Erster Pfad nach B



Start \ End	A	B	C	D	E	F	G
A	–					B	
B		–					
C			–				
D				–			
E					–		
F						–	
G							–

7 Tabelle des ersten Wegepunktes

- Die Tabelle für Knoten A als Startpunkt



Start \ End	A	B	C	D	E	F	G
A	—	B	B	B	B	B	B
B		—					
C			—				
D				—			
E					—		
F						—	
G							—

- Die Endtabelle

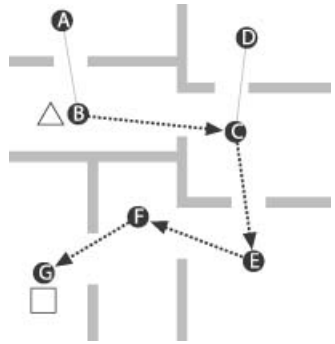
Start \ End	A	B	C	D	E	F	G
A	—	B	B	B	B	B	B
B	A	—	C	C	C	C	C
C	B	B	—	D	E	E	E
D	C	C	C	—	C	C	C
E	C	C	C	C	—	F	F
F	E	E	E	E	E	—	G
G	F	F	F	F	F	F	—

7 Wegfinden mit der Tabelle

- Wir suchen mit der Tabelle nun einen Pfad. Startpunkt→B Zielpunkt->G
- Schrittweise Pfadbestimmung
- Weg also: B->C->E->F->G

Start \ End	A	B	C	D	E	F	G
A	—	B	B	B	B	B	B
B	A	—	C	C	C	C	C
C	B	B	—	D	E	E	E
D	C	C	C	—	C	C	C
E	C	C	C	C	—	F	F
F	E	E	E	E	E	—	G
G	F	F	F	F	F	F	—

Goal B → G		
B → G	Table intersection → C	Move → C
C → G	Table intersection → E	Move → E
E → G	Table intersection → F	Move → F
F → G	Table intersection → G	Move → G



8 Zusammenfassung und Ausblick

- Wir haben gelernt:
- Grundlegende Verfahren (Basics, Obstacle Avoidance)
- Verfahren erstellt unter Mithilfe des Spielers oder eines Leaders (Breadcrumb Pathfinding)
- Verfahren mit dem Ziel auf der Strecke zu bleiben oder zu erforschen (Path Following, Wall Tracing)
- Verfahren mit Ressourcenschonung durch vorberechnete Wege (Waypoint Navigation)

- Komplexeres Verfahren: A* Pathfinding
- Unter anderem eingesetzt in Age of Empires und Enemy Nations
- nach diesem Vortrag

9 Counter-Strike

- Actionspiel, Urversion erschien Juni 1999
- Terroristen gegen Antiterroreinheit
- Antiterroreinheit soll (unter anderem) Geiseln retten
- Antiterroreinheit menschlich gesteuert
- Geisel vom Computer gesteuert folgen automatisch der Antiterroreinheit
- Pathfinding bei Geiseln->Line of Sight auf Antiterroreinheit

-> Bei kleinsten Hindernissen bleiben die Geiseln hängen

-> Läuft die Antiterroreinheit um die Ecke, bleiben die Geiseln an der Ecke hängen

9 Theseus

- Release 1.2.2006
- Miserables Pathfinding (und Grafik)
- LineOfSight

Anhang: Quellen

- AI for Game Developers (Bourg&Seeman), Chapter 6
- Theseus-Video: Gamestar 04/2006
- www.counter-strike.de
- http://gtr-game.10tacle.com/index.php?id=246&L=1 (GTR2 Screen)
- http://www.4players.de/4players.php/spielinfo/PC-CDROM/3031/Racing_Simulation_3.html (Racing Sim. Screen)
- http://gameswelt.de/pc/screenshots/detail.php?item_id=49639 (Civ4-Screen)
- http://www.richlandmaps.com/training/glossary/m_p.html (GIS-Glossary)
- http://en.wikipedia.org/wiki/Pathfinding (Wikipedia)
- www.gameai.com (allgemeine Informationen)

Fragen

?