

Informatik-Seminar (Spiele-KI)

Ermittlung eines guten Wegenetzes für Navigationsalgorithmen

Jens Remus

jens.remus@gmx.de

Technische Informatik



30. April 2007

1 Einleitung

- Motivation
- Begriffsdefinitionen
- Suchraum- / Spielweltrepräsentationen

2 Automatische Generierung von Navigation Meshes

- Eigenschaften „guter“ bzw. „optimaler“ Navigation Meshes
- Operationen auf konvexen Polygonen
- Generierung des Navigation Mesh aus der Spielweltgeometrie
- Optimierung der Datenstrukturen

3 Ausblick

- AI Game Programming Wisdom 4



1 Einleitung

- Motivation
- Begriffsdefinitionen
- Suchraum- / Spielweltrepräsentationen

2 Automatische Generierung von Navigation Meshes

- Eigenschaften „guter“ bzw. „optimaler“ Navigation Meshes
- Operationen auf konvexen Polygonen
- Generierung des Navigation Mesh aus der Spielweltgeometrie
- Optimierung der Datenstrukturen

3 Ausblick

- AI Game Programming Wisdom 4



Die Performanz der Spiele-KI Navigation ist abhängig von:

- Suchalgorithmus (*Best-First, Dijkstra, A**, ...)
- Suchraum- / Spielweltrepräsentation (Grids, Waypoints, *Navigation Meshes*, ...)

Vorhergehender Vortrag

Die Optimierung von Suchalgorithmen wurde im vorhergehenden Vortrag „Anpassungen des A*-Algorithmus an reale Anwendungen“ am Beispiel des *A* Algorithmus* von Jan Schliep erläutert.

Dieser Vortrag

Dieser Vortrag erläutert die Optimierung des Suchraums – der Repräsentation der Spielwelt für Navigationsalgorithmen am Beispiel der *Navigation Meshes*.



Die Performanz der Spiele-KI Navigation ist abhängig von:

- Suchalgorithmus (*Best-First, Dijkstra, A**, ...)
- Suchraum- / Spielweltrepräsentation (Grids, Waypoints, *Navigation Meshes*, ...)

Vorhergehender Vortrag

Die Optimierung von Suchalgorithmen wurde im vorhergehenden Vortrag „Anpassungen des A*-Algorithmus an reale Anwendungen“ am Beispiel des *A* Algorithmus* von Jan Schliep erläutert.

Dieser Vortrag

Dieser Vortrag erläutert die Optimierung des Suchraums – der Repräsentation der Spielwelt für Navigationsalgorithmen am Beispiel der *Navigation Meshes*.



Die Performanz der Spiele-KI Navigation ist abhängig von:

- Suchalgorithmus (*Best-First, Dijkstra, A**, ...)
- Suchraum- / Spielweltrepräsentation (Grids, Waypoints, *Navigation Meshes*, ...)

Vorhergehender Vortrag

Die Optimierung von Suchalgorithmen wurde im vorhergehenden Vortrag „Anpassungen des A*-Algorithmus an reale Anwendungen“ am Beispiel des *A* Algorithmus* von Jan Schliep erläutert.

Dieser Vortrag

Dieser Vortrag erläutert die Optimierung des Suchraums – der Repräsentation der Spielwelt für Navigationsalgorithmen am Beispiel der *Navigation Meshes*.

Vertex (latein: Punkt) Punkt, Vektor oder n-Tupel im Raum.
Ecke eines Polygons.

Polygon (griechisch: „Vieleck“) Gebiet einer Fläche, welche durch geschlossenen Streckenzug begrenzt ist.

konvexes Polygon Alle inneren Winkel des Polygons $\leq 180^\circ$.

Triangulation Verfahren zur Erzeugung eines Dreiecknetzes aus einer Punktmenge.



Begriffsdefinitionen

Polygon: konvex, konkav

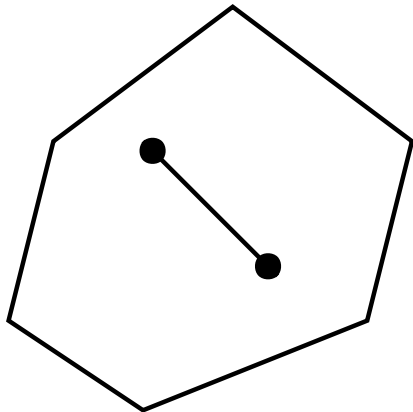


Abbildung: konvexes Polygon

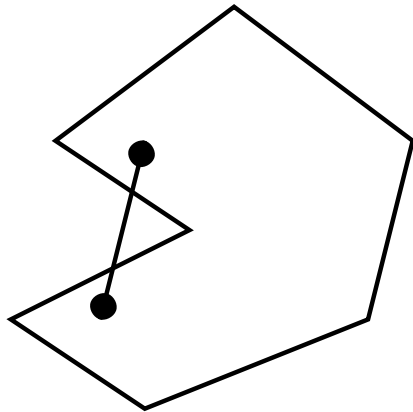


Abbildung: konkaves Polygon



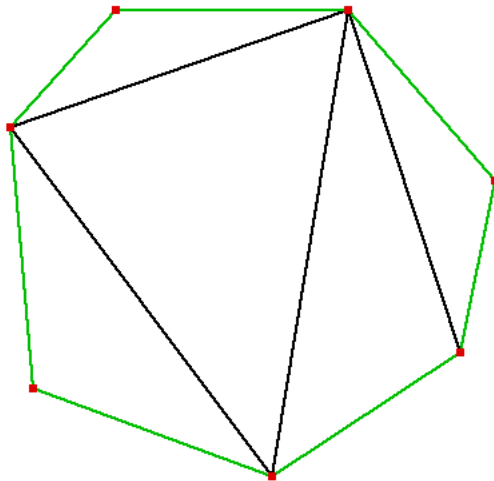


Abbildung: Delaunay-Triangulation einer Punktmenge

Alle Suchräume sind Graphen

Sie bestehen aus:

- Navigationsfeldern / -knoten, den Ecken
- Verbindungen zwischen den Feldern, den Kanten

Kriterien bei der Auswahl der Suchraumrepräsentation

- Performanz (Geschwindigkeit der ermöglichten Suchen) ↑
- Speicherbedarf ↓
- Bewegungsfähigkeiten der Agenten
- Generierung des Suchraums



Alle Suchräume sind Graphen

Sie bestehen aus:

- Navigationsfeldern / -knoten, den Ecken
- Verbindungen zwischen den Feldern, den Kanten

Kriterien bei der Auswahl der Suchraumrepräsentation

- Performanz (Geschwindigkeit der ermöglichten Suchen) ↑
- Speicherbedarf ↓
- Bewegungsfähigkeiten der Agenten
- Generierung des Suchraums



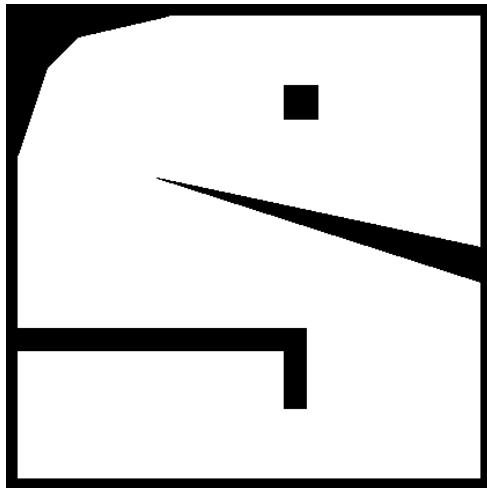


Abbildung: Beispiel-Spielwelt, *Pathfinding Demo*, Paul Tozour



Suchraum- / Spielweltrepräsentationen

Regular Grids (Reguläre Gitter): Square

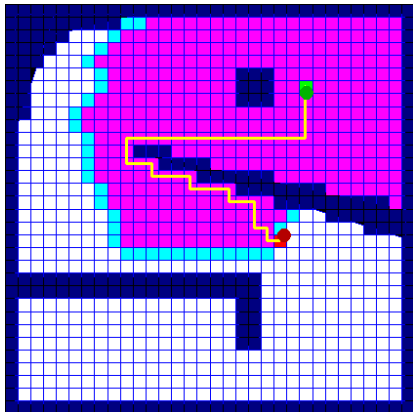


Abbildung: Square Cells, 4-Way

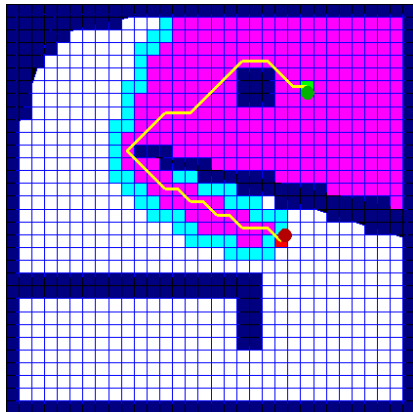


Abbildung: Square Cells, 8-Way



Suchraum- / Spielweltrepräsentationen

Regular Grids (Reguläre Gitter): Hexagonal

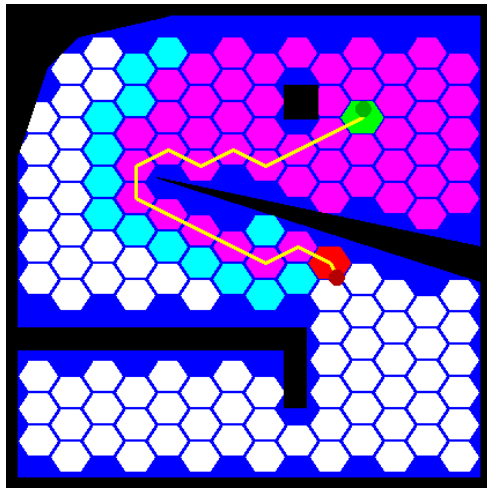


Abbildung: Hexagonal Cells



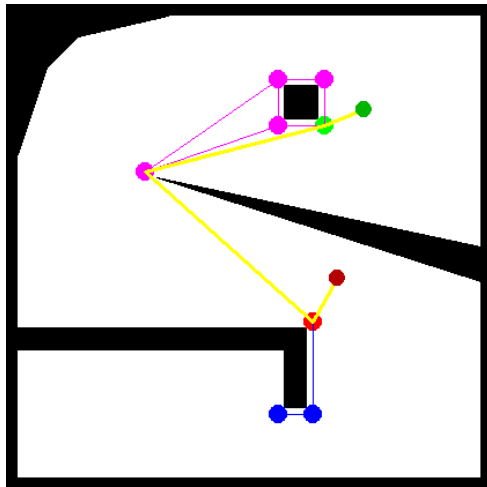


Abbildung: Corner Graph

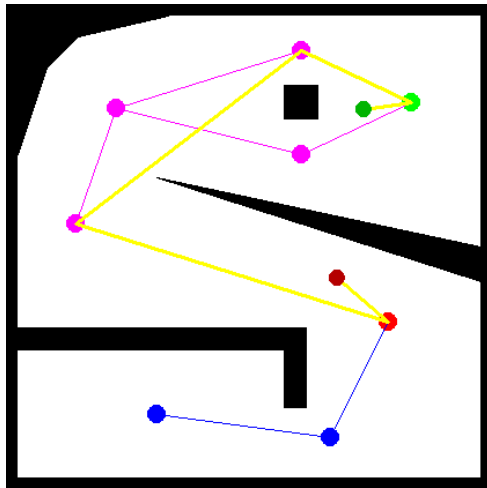


Abbildung: Waypoint Graph

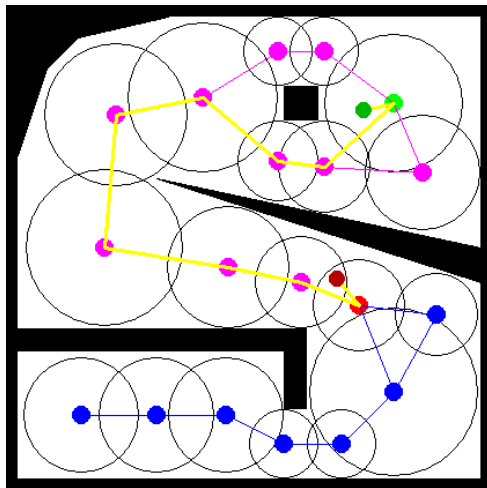


Abbildung: Circle-Based Waypoint Graph



Suchraum- / Spielweltrepräsentationen

Space-Filling Volumes

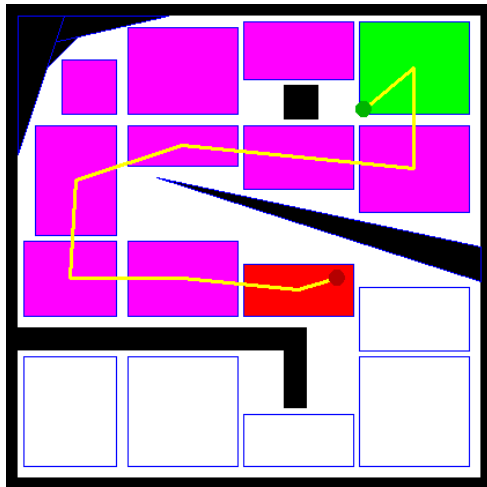


Abbildung: Space-Filling Volumes



Suchraum- / Spielweltrepräsentationen

Space-Filling Volumes: Beispiel

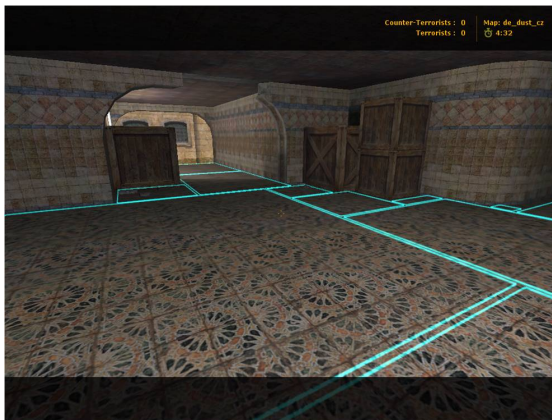


Abbildung: *Navigation Areas* in „Dust“, Counter-Strike:Source (Valve),
[Michael Booth, Game Developers Conference 2004]



Suchraum- / Spielweltrepräsentationen

Space-Filling Volumes: Beispiel

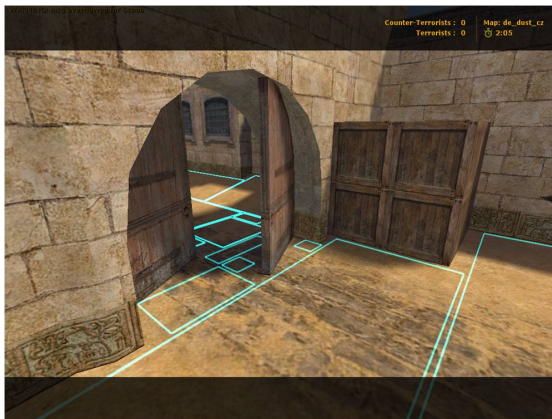


Abbildung: *Navigation Areas* in „Dust“, Counter-Strike:Source (Valve),
[Michael Booth, Game Developers Conference 2004]



Suchraum- / Spielweltrepräsentationen

Space-Filling Volumes: Beispiel

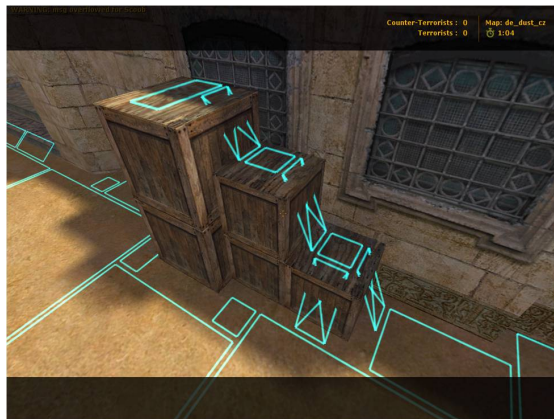


Abbildung: *Navigation Areas* in „Dust“, Counter-Strike:Source (Valve),
[Michael Booth, Game Developers Conference 2004]



Suchraum- / Spielweltrepräsentationen

Space-Filling Volumes: Beispiel

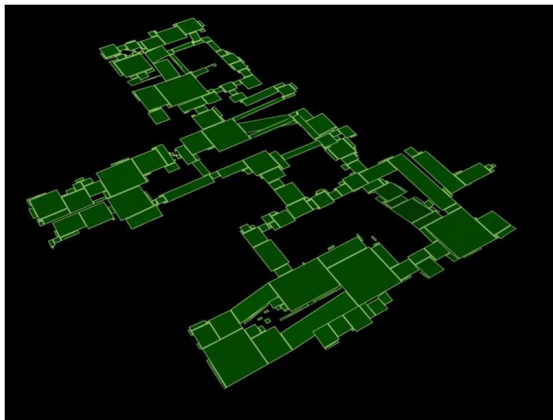


Abbildung: *Navigation Areas* in „Dust“, Counter-Strike:Source (Valve),
[Michael Booth, Game Developers Conference 2004]



Suchraum- / Spielweltrepräsentationen

Navigation Meshes

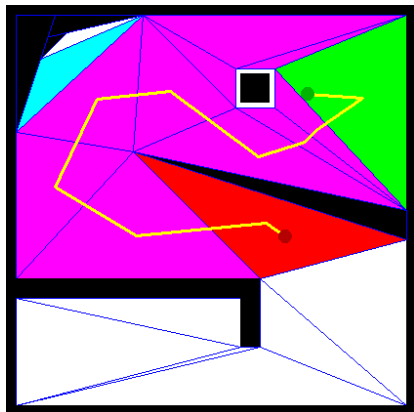


Abbildung: Navigation Mesh (NavMesh),
Triangle-Based

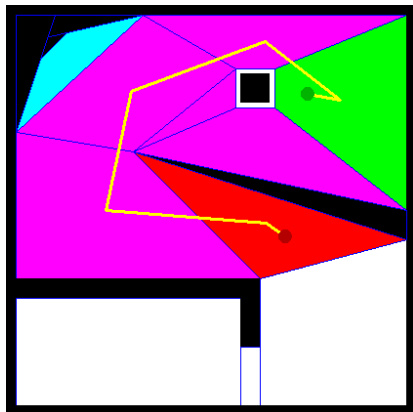


Abbildung: Navigation Mesh (NavMesh),
N-Sided-Poly-Based

Suchraum- / Spielweltrepräsentationen

Navigation Meshes: Beispiel

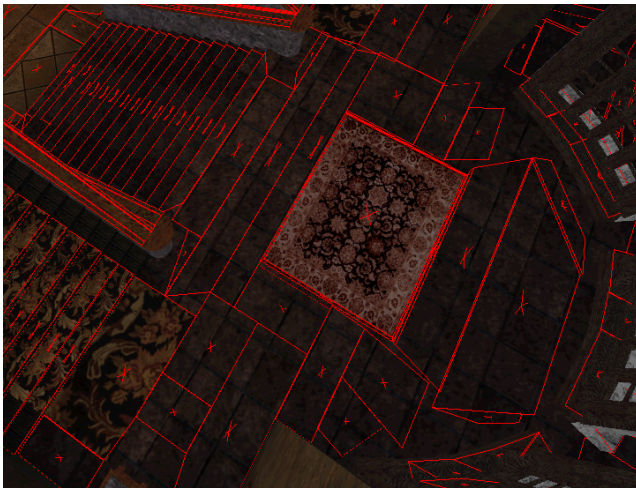


Abbildung: *Navigation Mesh*, [Paul Tozour, AI Game Programming Wisdom]



Suchraum- / Spielweltrepräsentationen

Navigation Meshes: Beispiel

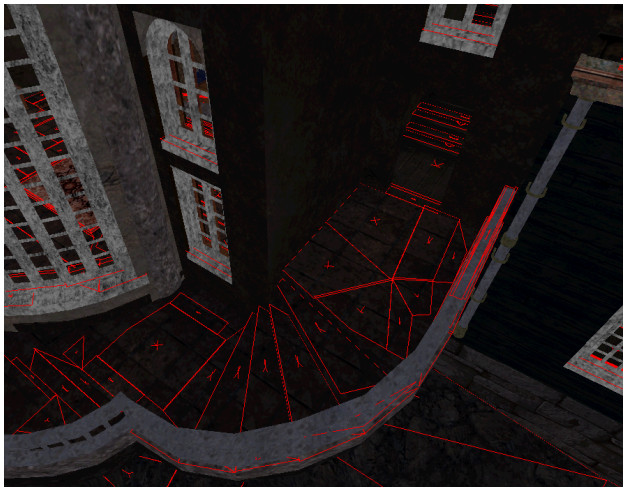


Abbildung: *Navigation Mesh*, [Paul Tozour, AI Game Programming Wisdom]



- 1 Einleitung
 - Motivation
 - Begriffsdefinitionen
 - Suchraum- / Spielweltrepräsentationen
- 2 Automatische Generierung von Navigation Meshes
 - Eigenschaften „guter“ bzw. „optimaler“ Navigation Meshes
 - Operationen auf konvexen Polygonen
 - Generierung des Navigation Mesh aus der Spielweltgeometrie
 - Optimierung der Datenstrukturen
- 3 Ausblick
 - AI Game Programming Wisdom 4



Das im folgenden erläuterte Verfahren basiert auf:

- „Building a Near-Optimal Navigation Mesh“
[Paul Tozour, AI Game Programming Wisdom]
- „Improving on Near-Optimality: More Techniques for Building Navigation Meshes“
[Fredrik Farnstrom, AI Game Programming Wisdom 3]



Eigenschaften eines „optimalen“ *Navigation Mesh*:

- Vollständigkeit
- Einfachheit
- Konsistenz
- Exzellente Laufzeit
- Vollständige Automatisierung
- Zumutbare Generationszeit
- Robust im Bezug auf Degenerationen
- Robust in der Handhabung von zusätzlicher einschneidender Geometrie

Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: 2 → 1 Merge (Hertel-Mehlhorn Algorithmus)

Hertel-Mehlhorn Algorithmus

- 1 Triangulation der Fläche.
- 2 Entfernung einer „unwichtigen“ Kante zwischen zwei Polygonen.
- 3 Wiederholung von Schritt 2 bis keine „unwichtigen“ Kante mehr existiert.



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: $2 \rightarrow 1$ Merge (Hertel-Mehlhorn Algorithmus)

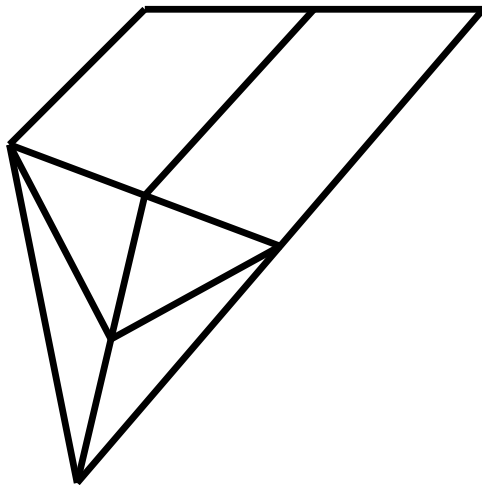


Abbildung: $2 \rightarrow 1$ Merge-Operation (Hertel-Mehlhorn Algorithmus)



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: $2 \rightarrow 1$ Merge (Hertel-Mehlhorn Algorithmus)

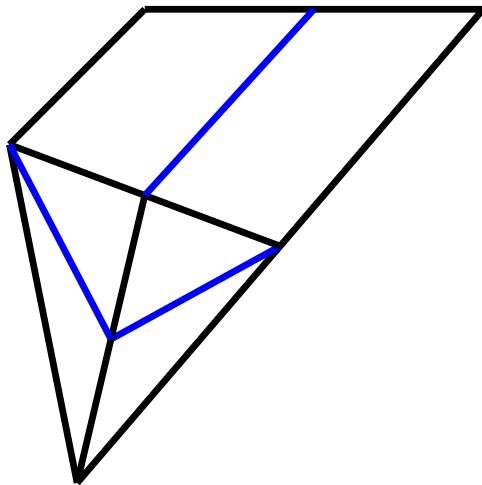


Abbildung: $2 \rightarrow 1$ Merge-Operation (Hertel-Mehlhorn Algorithmus)



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: $2 \rightarrow 1$ Merge (Hertel-Mehlhorn Algorithmus)

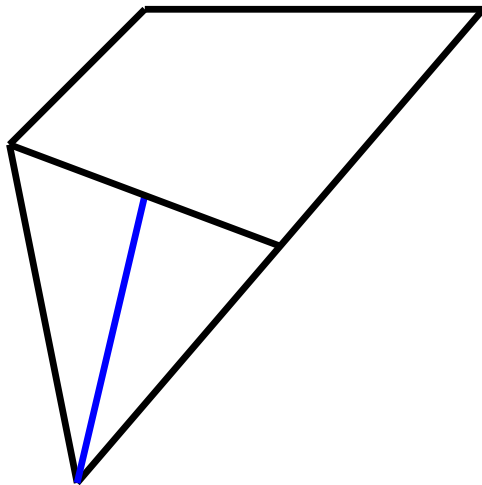


Abbildung: $2 \rightarrow 1$ Merge-Operation (Hertel-Mehlhorn Algorithmus)



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: $2 \rightarrow 1$ Merge (Hertel-Mehlhorn Algorithmus)

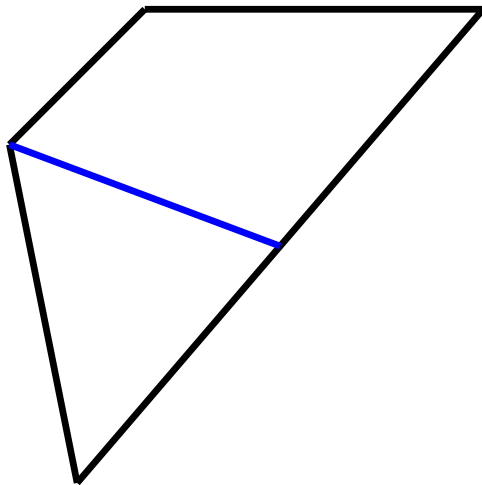


Abbildung: $2 \rightarrow 1$ Merge-Operation (Hertel-Mehlhorn Algorithmus)



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: $2 \rightarrow 1$ Merge (Hertel-Mehlhorn Algorithmus)

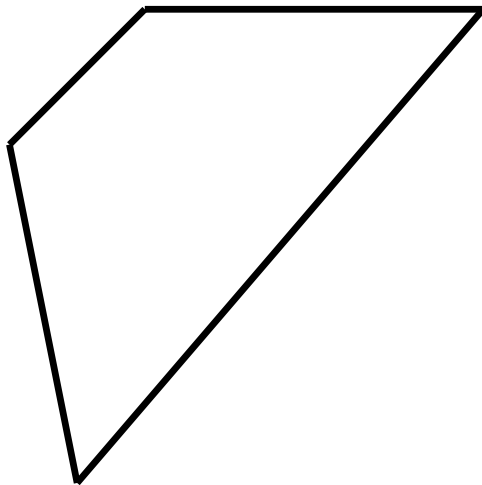


Abbildung: $2 \rightarrow 1$ Merge-Operation (Hertel-Mehlhorn Algorithmus)



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: $3 \rightarrow 2$ Merge

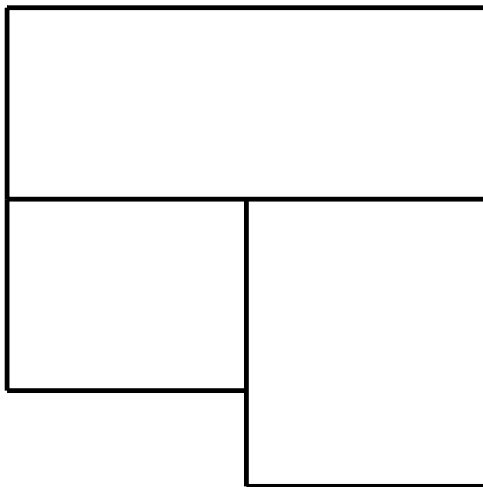


Abbildung: $3 \rightarrow 2$ Merge-Operation



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: $3 \rightarrow 2$ Merge

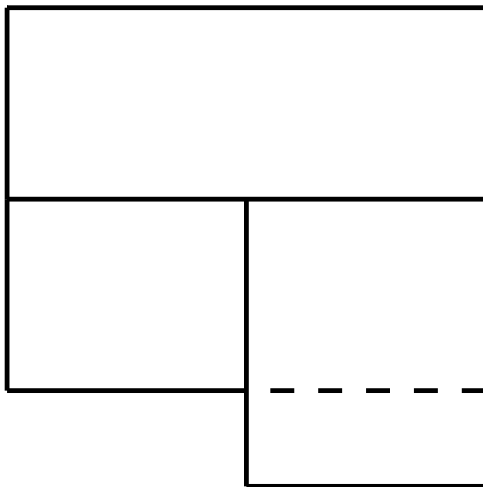


Abbildung: $3 \rightarrow 2$ Merge-Operation



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: $3 \rightarrow 2$ Merge

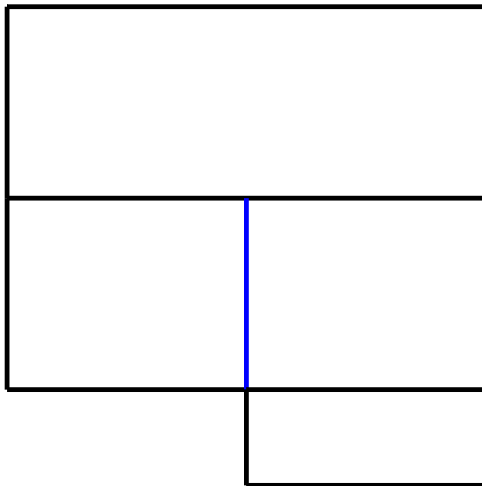


Abbildung: $3 \rightarrow 2$ Merge-Operation



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: $3 \rightarrow 2$ Merge

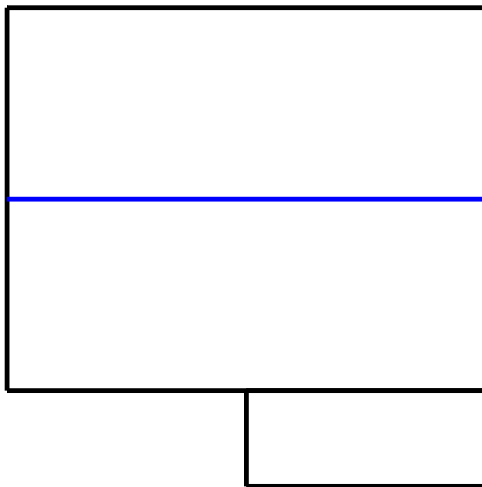


Abbildung: $3 \rightarrow 2$ Merge-Operation



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: $3 \rightarrow 2$ Merge

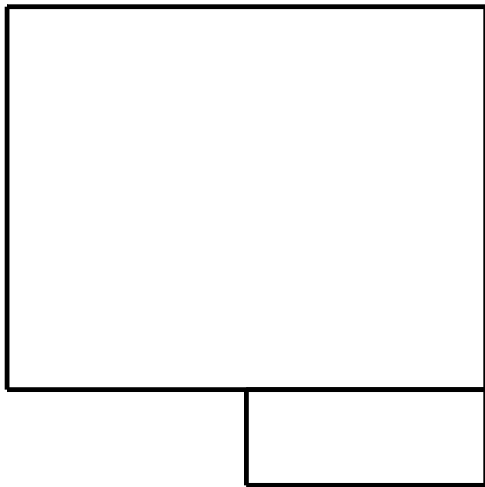


Abbildung: $3 \rightarrow 2$ Merge-Operation



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: $N \rightarrow 1$ Merge

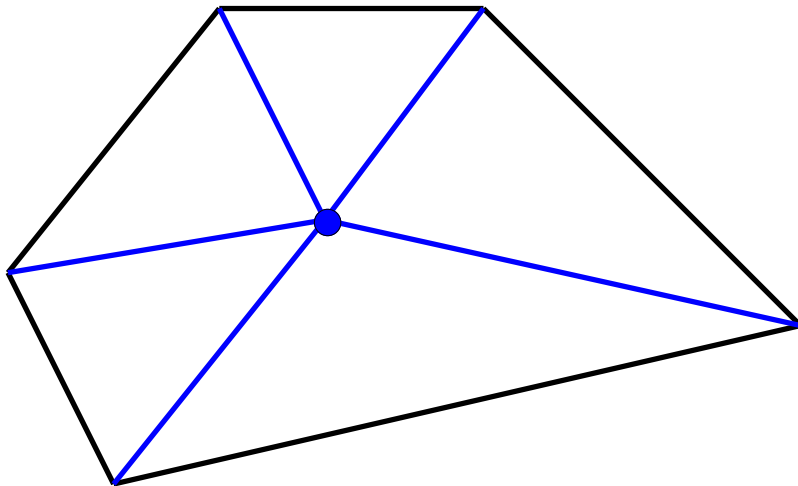


Abbildung: $N \rightarrow 1$ Merge-Operation



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: $N \rightarrow 1$ Merge

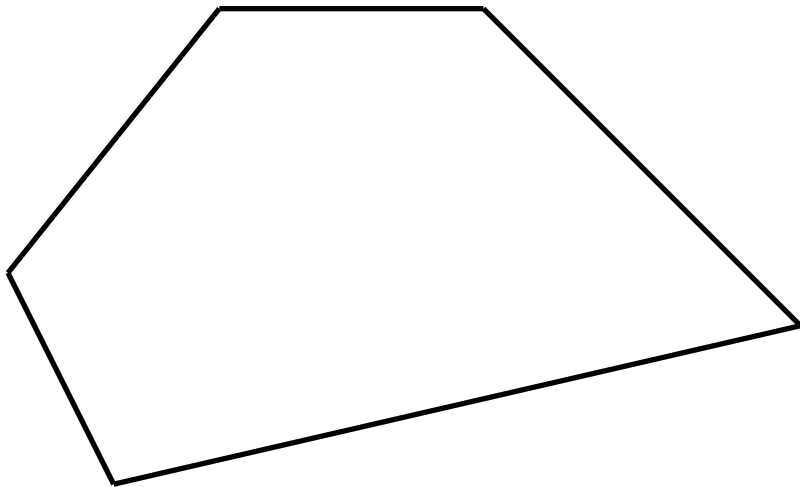


Abbildung: $N \rightarrow 1$ Merge-Operation



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: Subdivide

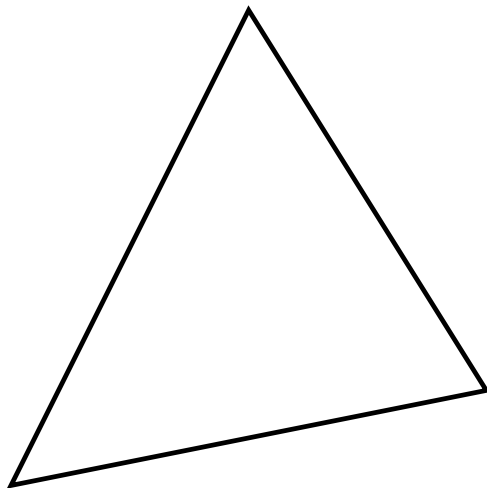


Abbildung: Subdivide-Operation, Dreieck und Viereck



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: Subdivide

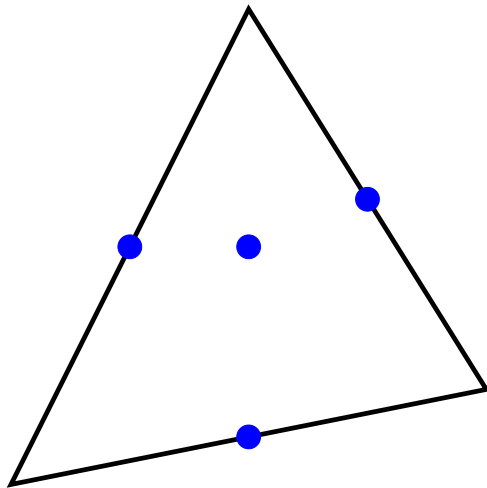


Abbildung: Subdivide-Operation, Dreieck und Viereck



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: Subdivide

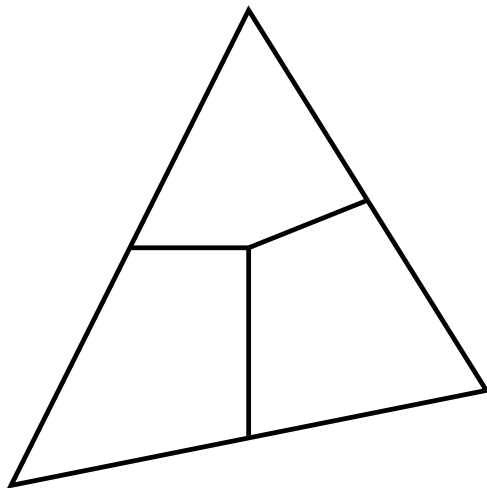


Abbildung: Subdivide-Operation, Dreieck und Viereck



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: Subdivide

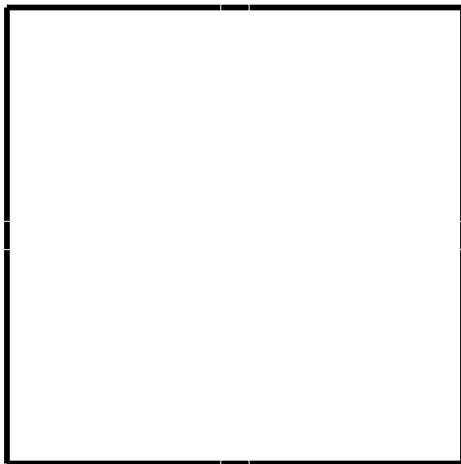


Abbildung: Subdivide-Operation, Dreieck und Viereck



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: Subdivide

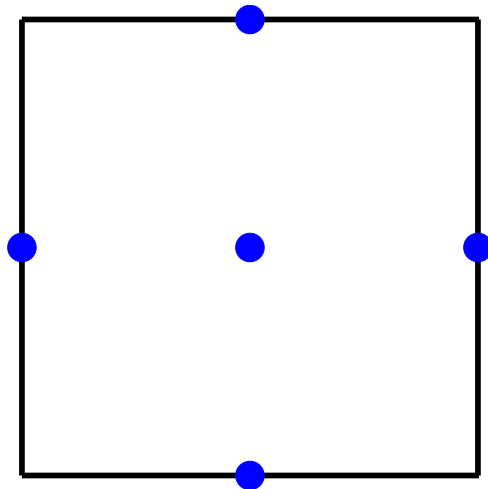


Abbildung: Subdivide-Operation, Dreieck und Viereck



Automatische Generierung von Navigation Meshes

Operationen auf konvexen Polygonen: Subdivide

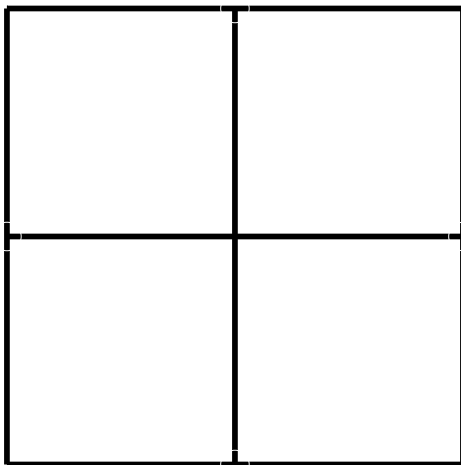


Abbildung: Subdivide-Operation, Dreieck und Viereck



Automatische Generierung von Navigation Meshes

Generierung des Navigation Mesh aus der Spielweltgeometrie

Vollständig automatische Generierung des *Navigation Mesh* aus der Spielweltgeometrie

- 1 Hinzufügen der Polygone der „begehbaren“ Spielweltgeometrie.
- 2 Wiederholte Anwendung der $2 \rightarrow 1$ Merge-Operation (Hertel-Mehlhorn).
- 3 Wiederholte Anwendung der $3 \rightarrow 2$ Merge-Operation.
- 4 Wiederholte Anwendung der $N \rightarrow 1$ Merge-Operation.
- 5 Wiederholung der Schritte 2, 3 und 4 bis keine weitere Optimierung mehr möglich.
- 6 Entfernung „unnützer“ *NavMesh*-Knoten.
- 7 Subtrahierung der statischen Hinderniss-Geometrie der Spielwelt (durch rekursive Anwendung der Subdivide-Operation).
- 8 Wiederholung der Schritte 2, 3 und 4 bis keine weitere Optimierung mehr möglich.



Normal-Pooling:

- Speichern aller Normalen-Vektoren in einem Pool.
- Polygon speichert einen Zeiger oder Index auf einen Normalen-Vektor anstelle des Vektors selbst.



Vertex-Pooling:

- Speichern aller Vertices in einem Vertex-Pool (Hash Map).
- Polygon speichert Liste von 2- oder 4-Byte Indices auf Vertices anstelle der Vertices selbst.



Automatische Generierung von Navigation Meshes

Optimierung der Datenstrukturen: Schnelle Bestimmung der direkten Nachbarschaft von Polygonen

Schnelle Bestimmung der direkten Nachbarschaft von Polygonen:

- Erweiterung des Vertex-Pooling.
- Vertex speichert Liste von Zeigern auf Polygone, die ihn verwenden.
- Suche aller benachbarten Polygone durch iterieren über Vertex-Pool.
- Suche der benachbarten Polygone zu einem bestimmten Polygon über dessen Vertices.



- 1 Einleitung
 - Motivation
 - Begriffsdefinitionen
 - Suchraum- / Spielweltrepräsentationen
- 2 Automatische Generierung von Navigation Meshes
 - Eigenschaften „guter“ bzw. „optimaler“ Navigation Meshes
 - Operationen auf konvexen Polygonen
 - Generierung des Navigation Mesh aus der Spielweltgeometrie
 - Optimierung der Datenstrukturen
- 3 Ausblick
 - AI Game Programming Wisdom 4



Kapitel: Pathfinding

- Team AI (Edmond Prakash)
- Memory Efficient Pathfinding Abstractions (Nathan Sturtevant)
- Fast Pathfinding Based on Triangulation Abstraction (Michael Buro)
- Real-Time Dynamic NavMesh Generation (Paul Marden, Forrest Smith)
- Automatic Generation of Path Nodes for a General Purpose 3D Environment (John Ratcliff)
- Intrinsic Detail in Navigation Mesh Generation (James Stewart, Colt McAnlis)
- NavMesh Generation: An Empirical Approach (David Hamm)
- Navigation Graph Generation in Highly Dynamic World (Ramon Axelrod)
- Path Planning in Dynamic Environments (Ferns Paanakker)
- Practical Path Finding in Dynamic Environments (Per-Magnus Olsson)
- Post Processing for High Quality Turns (Chris Journey)

Annähernd 50% der Artikel des Kapitels Pathfinding handeln von *Navigation Meshes*.



Kapitel: Pathfinding





- Team AI (Edmond Prakash)
- Memory Efficient Pathfinding Abstractions (Nathan Sturtevant)
- Fast Pathfinding Based on Triangulation Abstraction (Michael Buro)
- Real-Time Dynamic NavMesh Generation (Paul Marden, Forrest Smith)
- Automatic Generation of Path Nodes for a General Purpose 3D Environment (John Ratcliff)
- Intrinsic Detail in Navigation Mesh Generation (James Stewart, Colt McAnlis)
- NavMesh Generation: An Empirical Approach (David Hamm)
- Navigation Graph Generation in Highly Dynamic World (Ramon Axelrod)
- Path Planning in Dynamic Environments (Ferns Paanakker)
- Practical Path Finding in Dynamic Environments (Per-Magnus Olsson)
- Post Processing for High Quality Turns (Chris Journey)

Annähernd 50% der Artikel des Kapitels Pathfinding handeln von *Navigation Meshes*.



Vielen Dank für die Aufmerksamkeit!

Fragen?

-  **AI Game Programming Wisdom 1-3**
Steve Rabin, Charles River Media
-  **Game Programming Gems 1-6**
Mark DeLoura / Dante Treglia / Andrew Kirmse / Kim Pallister /
Mike Dickheiser, Charles River Media
-  **Fredrik Farnstrom—Rockstar, San Diego**
Near-Optimality: More Techniques for Building Navigation Meshes
AI Game Programming Wisdom 3, Charles River Media, 2006,
Kapitel 2.2, S. 113-128
-  **Paul Tozour—Ion Storm Austin**
Building a Near-Optimal Navigation Mesh
AI Game Programming Wisdom, Charles River Media, 2002,
Kapitel 4.3, S. 171-185



Paul Tozour—Retro Studios

Search Space Representations

AI Game Programming Wisdom 2, Charles River Media, 2003,
Kapitel 2.1, S. 85-102



Michael Booth—Turtle Rock Studios

The Making of the Official Counter-Strike Bot

Game Developers Conference 2004,
<http://www.gdconf.com/conference/2004.htm>