



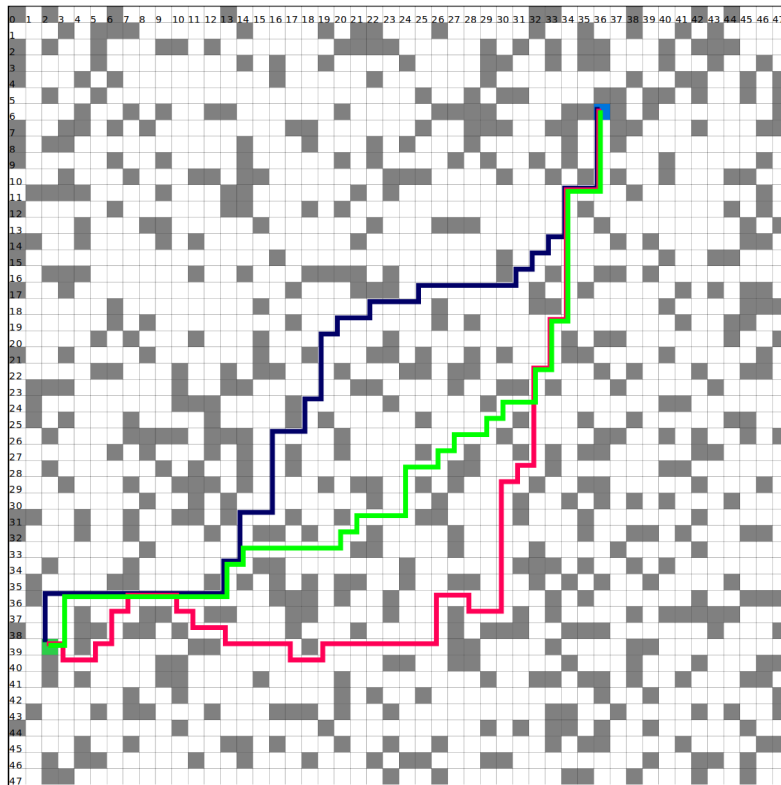
BERUFSMATURITÄTSSCHULE ZÜRICH

BERUFSMATURITÄTSARBEIT

TECHNIK, ARCHITEKTUR, LIFE SCIENCES

Pathfinding-Algorithmen: Einführung und Vergleich mittels einer Webapplikation

Oberthema: MOBILITÄT



SEVERIN FÜRBRINGER

severin@fsfe.org

EVT18a

ADRIAN STOOP

adrian-stoop@gmx.ch

EVT18a

DR. JÜRG PÖTTINGER

Begleitperson

1. Februar, 2018

Abstract

Diese Berufsmaturitätsarbeit behandelt folgende Fragestellung: Was sind die technischen Eigenschaften und Vor- und Nachteile von drei ausgewählten Pathfinding-Algorithmen und wie unterscheiden sie sich? Um diese Fragestellung zu beantworten, wird im Rahmen dieser Arbeit eine Webapplikation entwickelt, die den Benutzer in das Thema einführt und Vergleiche zwischen drei Pathfinding-Algorithmen interaktiv, parallel und mehrmals hintereinander ermöglicht. Um den Vergleich möglich zu machen, generiert die Webapplikation automatisch Raster mit zufällig platzierten Wänden. Die Bedingungen (Parameter) des Vergleichs können vom Benutzer auch angepasst werden. Die experimentartige Webapplikation liefert für statistische Zwecke Messwerte, welche dann für die Ableitung der Vor- und Nachteile der drei Pathfinding-Algorithmen miteinbezogen werden.

Die Webapplikation ist unter `bma.fuerbringer.info` zugänglich. Der dazugehörige Quellcode ist frei unter `github.com/fuerbringer/bma` erhältlich.

Inhaltsverzeichnis

1	Einleitung	2
2	Realisierung	3
2.1	Konzept	3
2.1.1	Pathfinding.js	3
2.1.2	Programmierwerkzeuge	3
2.1.3	Konzipierung der Benutzeroberfläche	4
2.2	Implementierung des Vergleichers	7
2.2.1	Raster	8
2.2.2	Auswertung und Merkmale	12
3	Anhang	15
3.1	Screenshots der Webapplikation	15
3.1.1	Pathfinder-Vergleicher	15
3.2	Danksagung	16
3.2.1	Schriftsetzung	16
4	Glossar	17

1. Einleitung

2. Realisierung

Dieser Abschnitt soll dazu verhelfen, die technischen Entscheidungen und Schritte zu erläutern und es möglich zu machen, die Kernfunktionen des Projekts zu reproduzieren. Leser, die diese Applikation, oder Teile davon, anderweitig einsetzen möchten, beziehen sich zusätzlich auf den frei verfügbaren Quellcode[3] des gesamten Projekts.

2.1 Konzept

Die Konzipierung dieser Webapplikation verlangte einige technische und gestalterische Entscheidungen. Einerseits musste die Programmiersprache und Struktur der Applikation geplant werden und andererseits auch das eigentliche Aussehen der Benutzeroberfläche der Webapplikation. Auf diese zwei wesentlichen Aspekte wird in diesem Abschnitt eingegangen.

2.1.1 Pathfinding.js

Eine korrekte Implementation der ausgewählten Pathfinder sind Voraussetzung für das Projekt. Daher wurde, um diese Voraussetzung zu erfüllen, die frei verfügbare Implementation PathFinding.js [1] verschiedener Pathfinder ausgewählt.

2.1.2 Programmierwerkzeuge

Die vorhandenen Fachkenntnisse und Erfahrungen schlugen für die Implementierung entweder PHP oder JavaScript als mögliche Programmiersprachen vor. Zum Entscheid massgebend war, dass die Programmbibliothek PathFinding.js [1], welche in unserer Arbeit eine wichtige Rolle spielt, in JavaScript implementiert wurde. Durch den Einsatz von JavaScript im Front-End und im Back-End, wäre es möglich die Pathfinder Berechnungen beliebig auf dem Rechner des Nutzers und auch auf dem Rechner des Servers auszuführen. Aus diesen Gründen wurde für das Front- und Back-End JavaScript ausgewählt. Folgend werden die wichtigsten Technologien unserer Webapplikation aufgelistet¹.

Programmiersprache JavaScript mit Einbindung von PathFinding.js [1]

¹Eine komplette Auflistung inklusive der abhängenden Programmbibliotheken macht wenig Sinn, vor allem da unser Quellcode frei ersichtlich ist.

Webserver-Software Express² übernimmt die Ausführung von JavaScript im Back-End.

Quellcode-Hosting GitHub³. Der Quellcode unser Webapplikation ist frei unter <https://github.com/fuerbringer/bma> zugänglich.

Webhosting Die Vultr⁴-Cloud dient als Hosting-Provider. Prinzipiell kann die Webapplikation jedoch auf beliebigen Unix und GNU/Linux Servern gehostet werden.

2.1.3 Konzipierung der Benutzeroberfläche

Als Webapplikation muss unsere Arbeit deren Nutzen dem Benutzer ausreichend klar machen. Umsomehr ist dies von Bedeutung, da die Webapplikation frei zugänglich ist und daher auch Besucher die Idee der Webapplikation verstehen sollten. Aus diesem Grund wurde unsere Webapplikation für den Nutzer einführend gestaltet. Das heisst, dass der Benutzer beim Aufruf der Webapplikation zunächst auf einer Willkommenseite landet, die ihn wiederum zu einer Einführung in das Thema und Ziel der Arbeit führt. Nach der Einführung kann der Benutzer in der Visualisierung mit einzelnen Pathfindern erste Erfahrungen machen. Im Abschluss kommt der Benutzer zum Hauptteil der Arbeit, dem Pathfinding-Vergleicher. Ziel dieser Struktur ist, dass jede Person wenigstens einen Einblick in die Welt der Pathfinding-Algorithmen erhält und über das nötige Wissen verfügt, das Ziel des Pathfinding-Vergleichers zu verstehen.

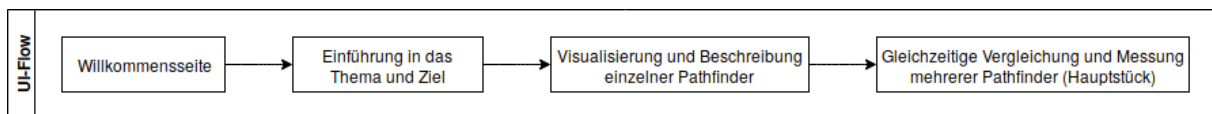


Abbildung 2.1: Webapplikationsstruktur. Quelle: Eigenleistung

²Express Webserver für Node Webapplikationen, <https://expressjs.com/>, Stand: 28. Januar 2019

³GitHub, <https://github.com/>, Stand: 28. Januar 2019

⁴Vultr - The Infrastructure Cloud™, <https://vultr.com/>, Stand: 28. Januar 2019

Einführungs- und Willkommenseite

Die Einführungsseite erklärt dem Benutzer was Algorithmen sind und führt ihn anschliessend mit einfachen Beispielen in das Thema der Pathfinder ein. Zuletzt werden die Ziele der Arbeit aufgelistet.

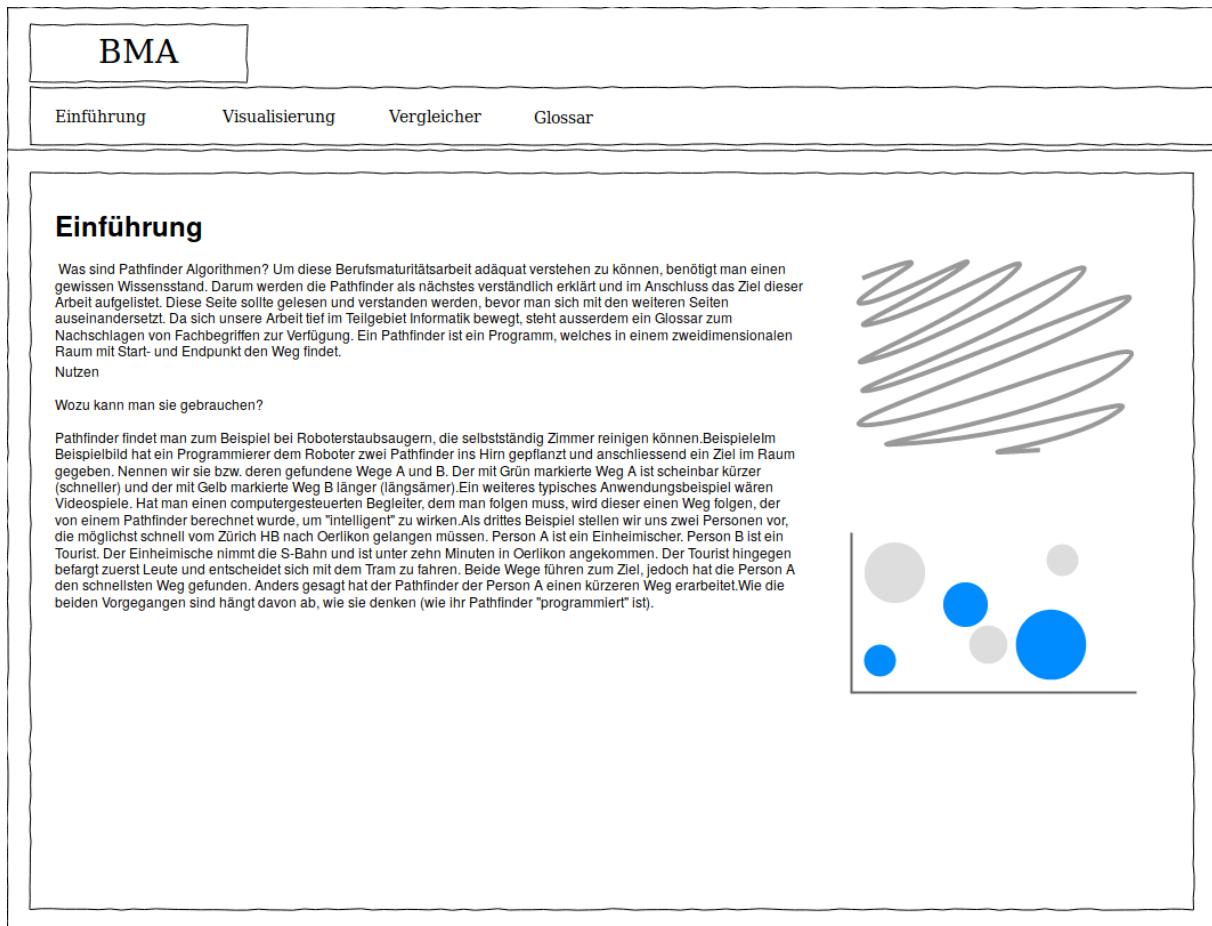


Abbildung 2.2: Einführungsseite. Quelle: Eigenleistung

Visualisierung der Pathfinder

Nach der Einführung erhält der Benutzer die Möglichkeit mit einzelnen Pathfindern zu experimentieren. Dazu kann er auch verschiedene Parameter anpassen. Dies ist die Vorstufe zum Pathfinder-Vergleich, da hier nochmals die einzelnen Pathfinder auf ihre Eigenschaften beschrieben werden.

BMA

EinführungVisualisierungVergleichGlossar

Visualisierung der Pathfinder

Auf der rechten Seite sieht man die in der Einführung erwähnten Pathfinder «in action». Der grüne Block repräsentiert jeweils den Endpunkt und der blaue den Startpunkt. Der ausgewählte Pathfinder findet zwischen diesen zwei Punkten dann den passenden Weg.

Diese Seite soll zur kurzen visuellen Einführung in die Pathfinder dienen. Parameter lassen sich vom Nutzer unter «Steuerung» anpassen.

A*-Pathfinder Beschreibung

...

BestFirstFinder Beschreibung

...

BreadthFirstFinder Beschreibung

...

Nächster Schritt

Im nächsten Schritt wird's interessant. Wir vergleichen die Pathfinder nebeneinander auf ihre Eigenschaften.

Raster

Messungen

Zurückgelegter Weg: A
Operationen: B
Rechenzeit: C

Parameter

Pathfinder-Art	<input type="text" value="A*"/>	
Heuristik	<input type="text" value="Manhattan"/>	
Rastertyp	<input type="text" value="Zufallsraster"/>	
Rastergrösse	<input type="text" value="Länge = 10"/>	<input type="text" value="Höhe = 8"/>
Diagonale	<input type="text" value="Nein"/>	

Vergleiche Ausführen

Abbildung 2.3: Benutzeroberflächenkonzept Visualisierer. Quelle: Eigenleistung

Vergleich der Pathfinder

Als Herzstück der Arbeit ermöglicht der Pathfinder-Vergleicher dem Nutzer die in unserer Arbeit ausgewählten Pathfinder zu vergleichen. Hier werden, gleich wie beim Visualisierer, zuerst vom Benutzer die Parameter gewählt. Die parallel ausgeführten Resultate mit Informationen über die Rechenzeit, zurückgelegten Weg und Operationen, sind dann interaktiv anwählbar. Gleichzeitig wird auch unter “Resultate” unten links das Total aller Vergleiche akkumuliert, damit ein Gesamteindruck verschaffen werden kann. Die Resultate dieser Seite dienen im Statistikteil als Datenquelle.

BMA

Einführung Visualisierung Vergleicher Glossar

Vergleich der Pathfinder

Pathfinding-Algorithmen werden hier verglichen (Side-by-side).

In der Einführung haben wir bereits die drei ausgewählten Pathfinder (A*, BestFirst und BreadthFirst) genannt. Diese Pathfinder werden auf dieser Seite parallel und mehrmals hintereinander verglichen. Als Resultat ersichtlich sind dann folgende Eigenschaften:

- Anzahl Operationen
- Zurückgelegter Weg
- Rechenzeit

Visuell ersichtlich sind die Durchläufe. Die Wege der jeweiligen Pathfinder sind verschieden gefärbt. Vom Nutzer anpassbar sind als Parameter die Anzahl Durchläufe, Rasterart und Rastergrösse.

Resultate

Anzahl Operationen: ...

Zurückgelegter Weg: ...

Vergangene Rechenzeit: ...

Raster

Resultate der Messungen: ...

Parameter

Anzahl Durchläufe

Rasterart

Rastergrösse

Vergleiche Ausführen

Abbildung 2.4: Benutzeroberflächenkonzept Vergleich. Quelle: Eigenleistung

2.2 Implementierung des Vergleichers

Vor allem für die statistischen Auswertungen ist es von Bedeutung zu wissen, wie der Vergleich zu den Resultaten kommt und die Berechnungen durchführt. Um an die statistischen Merkmale zu gelangen, musste je nach Merkmal die Webapplikation oder PathFinding.js erweitert werden.

2.2.1 Raster

Pathfinder benötigen einen Raum, um ihre Arbeit zu verrichten. Folglich wird erläutert, wie diese Räume im Quellcode der Arbeit implementiert wurden. Die Implementation der Pathfinding-Algorithmen von PathFinding.js erwartet als Raum eine Liste mit beispielsweise 8×8 Zellen:

```
[(0,0), (1,0), (2,0), (3,0), (4,0), (5,0), (6,0), (7,0)]
[(0,1), (1,1), (2,1), (3,1), (4,1), (5,1), (6,1), (7,1)]
[(0,2), (1,2), (2,2), (3,2), (4,2), (5,2), (6,2), (7,2)]
[(0,3), (1,3), (2,3), (3,3), (4,3), (5,3), (6,3), (7,3)]
[(0,4), (1,4), (2,4), (3,4), (4,4), (5,4), (6,4), (7,4)]
[(0,5), (1,5), (2,5), (3,5), (4,5), (5,5), (6,5), (7,5)]
[(0,6), (1,6), (2,6), (3,6), (4,6), (5,6), (6,6), (7,6)]
[(0,7), (1,7), (2,7), (3,7), (4,7), (5,7), (6,7), (7,7)]
```

Zu beachten ist, dass die obige Liste pro Element (Zelle) lediglich die Koordinate (x,y) als Eigenschaft aufweist. Weitere Eigenschaften pro Zelle, wie die Wandeigenschaft, werden in späteren Abschnitten erwähnt. Diese Art von Raster ist bei der Darstellung von Pixelmatrizen und sonstigen computergrafikorientierten Anwendungen üblich. Man kann sich diese Anordnung auch als den vierten Quadranten eines kartesischen Koordinatensystems vorstellen, dessen y -Koordinate ihren Absolutwert annimmt. Des Weiteren gilt $x \in \mathbb{N}$ und $y \in \mathbb{N}$. Daraus folgt, dass Zwischenschritte nicht möglich sind, beispielsweise das Ziel nicht die Koordinate $(\frac{1}{2}, \frac{42}{11})$ annehmen kann. Ein weiteres Beispiel dafür wären die Pixel eines gewöhnlichen Monitors, denn einen Bruchteil eines Pixels anzusteuern ist ebenfalls nicht möglich. Benötigt man eine derartige Präzision, erhöht man dazu am besten die Auflösung des Raums. Es folgt eine Abbildung eines zweidimensionalen leeren Raumes der oben gegebenen Liste.

0	1	2	3	4	5	6	7
1							
2							
3							
4							
5							
6							
7							

Abbildung 2.5: Ein von unserer Webapplikation generiertes 8×8 Raster. Quelle: Eigenleistung

Generierung der Wände und Gänge

Damit die Eigenschaften der Pathfinder besser zum Vorschein kommen, sie also nicht nur im leeren Raum ausgeführt werden, wurde im nächsten Schritt für den Rastergenerator die Logik zum Verstreu von Wänden implementiert. Die Rastermatrix erhält dann pro Zelle die Eigenschaft, ob sie vom Pathfinder passierbar ist oder nicht. Die Programmbibliothek PathFinding.js versteht diese Logik genauso, indem man zu einer gegebenen Koordinate definiert, ob sie eine Wand ist. Der folgende prozedurale Pseudocode definiert die in unserer Webapplikation benutzte Routine⁵, um Raster mit Wänden zu füllen.

```

1: procedure WANDGENERATOR(raster, freq)
2:   for  $y \leftarrow 0$  to count(Zeilen von raster) do
3:     for  $x \leftarrow 0$  to count(Zellen von raster in der Zeile y) do
4:        $rand \leftarrow random(0, 100)$  ▷ Zufallszahl  $[0, 100]$ 
5:        $x \leftarrow \lfloor rand \bmod freq \rfloor$  ▷ Rest von  $\frac{rand}{freq}$ 
6:       if  $x = 0$  then
7:         Zelle in raster bei  $x$  und  $y$  wird zu einer Wand
8:   return raster

```

Vom Frequenzparameter *freq* hängt schlussendlich ab, wie oft Wände im Raster vorkommen. Er kann frei gewählt werden. Nähert sich der Wert *freq* jedoch 1 mit $freq > 1$, erhöht sich auch die Anzahl unlösbarer Raster, da immer weniger mögliche Wege zwischen den verbliebenen Zellen besteht. Wendet man den Wandgenerator in unserem Raster der Abbildung 2.5 an, so erhält man ein Raster mit Wänden.

⁵Die genaue Codestelle in der Webapplikation ist hier ersichtlich: <https://github.com/fuerbringer/bma/blob/master/src/maze.js#L8>

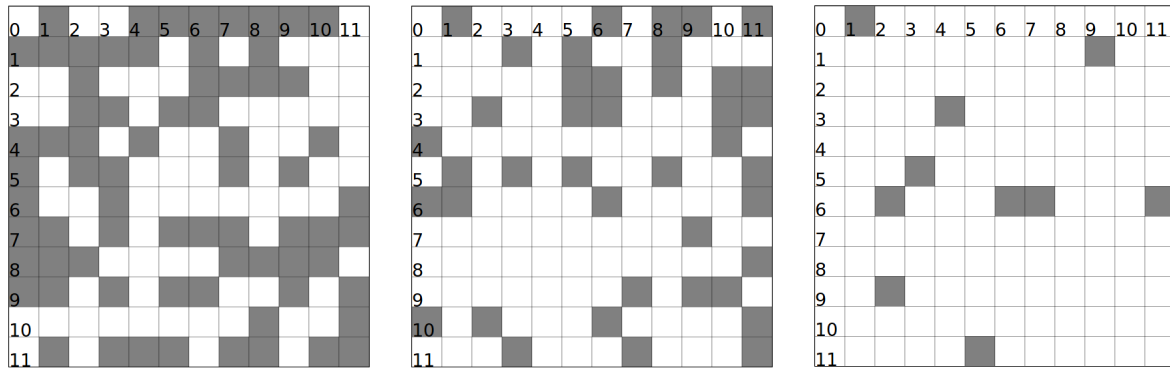


Abbildung 2.6: Raster mit Frequenzparametern $freq = 2$, $freq = 4$ und $freq = 16$ im Wandgenerator. Quelle: Eigenleistung

Wir sehen also, dass mit dem Frequenzparameter $freq$ die Häufigkeit der unpassierbaren Wände geändert werden kann. In unserer Webapplikation haben wir für den Frequenzparameter $freq = 4$ ausgewählt, da dieser Wert ein gutes Gleichgewicht zwischen nicht zu viel unlösbaren Rastern und genügend Wänden ergibt.

Generierung des Start- und Endpunkts

Wählt man zwei zufällige passierbare Zellen als Start- und Endpunkt aus, erhält man ein Raster, mit dem man Pathfinding-Algorithmen anwenden kann. In unserer Webapplikation werden diese zwei Zellen farbig differenziert. Programmatikalisch handelt es sich um Zellen, die Wände sind und in Ziel- resp. Endpunkte umgewandelt werden. Die folgenden zwei Routinen⁶ wählen in einem gegebenen Raster einen Start- und einen Endpunkt aus.

```

1: procedure STARTUNDENDPUNKTGENERATOR(raster)
2:   frei ist eine leere Liste                                ▷ mögliche Zellen für den Start- und Endpunkt
3:   for y ← 0 to count(Zeilen von raster) do
4:     for x ← 0 to count(Zellen von raster in der Zeile y) do
5:       zelle ← Zelle mit (x,y) in raster
6:       if zelle ist keine Wand then
7:         zelle der Liste frei hinzufügen
8:   raster ← PUNKTEMARKIEREN(raster, frei)
9:   return raster

10: procedure PUNKTEMARKIEREN(raster, frei)                  ▷ Start- und Endpunkte markieren
11:   startIndex ←  $\lfloor \text{random}(0, \text{count}(\textit{frei})) \rfloor$       ▷ Zufallsindex für die Liste frei
12:   Let
13:     sx ← x-Koordinate von frei am Index startIndex
14:     sy ← y-Koordinate von frei am Index startIndex
15:   In Zelle in raster bei sx und sy wird zum Startpunkt

```

⁶Die genaue Codestelle in der Webapplikation ist hier ersichtlich: <https://github.com/fuerbringer/bma/blob/master/src/maze.js#L27>

```

16:   Element in frei am Index startIndex entfernen
17:   endIndex  $\leftarrow \lfloor \text{random}(0, \text{count}(\text{frei})) \rfloor$  ▷ Zufallsindex für die Liste frei
18:   Let
19:     ex  $\leftarrow$  x-Koordinate von frei am Index endIndex
20:     ey  $\leftarrow$  y-Koordinate von frei am Index endIndex
21:   In Zelle in raster bei ex und ey wird zum Endpunkt
22:   Element in frei am Index endIndex entfernen
23:   return raster

```

Wendet man nun den Start- und Endpunktgenerator⁷ an Rastern mit Wänden (siehe Abbildung 2.6) an, so erhält man folgendes, ein für den PathFinder brauchbares, Raster mit Wänden und Start- und Endpunkt:



Abbildung 2.7: Ein von unserer Webapplikation generiertes 8×8 Raster mit Wänden und markierten Start- und Endpunkten. Quelle: Eigenleistung

Im letzten Schritt lässt man die PathFinder den Weg zwischen den markierten Start- und Endpunkten finden. Die Parameter, die man PathFinding.js dafür übergeben muss, sind grundsätzlich das Raster und die darin markierten Punkte, die als Start- und Ende dienen. Die markierten Punkte speichert man entweder im Start- und Endpunktgenerator zwischen oder man benutzt eine weitere Routine⁸, die die Start- und Endpunkte aus einem Raster extrahiert:

```

1: procedure STARTUNDENDPUNKTFINDEN(raster)
2:   startPunkt  $\leftarrow$  leeres Element
3:   endPunkt  $\leftarrow$  leeres Element

```

⁷Die genaue Codestelle in der Webapplikation ist hier ersichtlich: <https://github.com/fuerbringer/bma/blob/master/src/maze.js#L27>

⁸Die genaue Codestelle in der Webapplikation ist hier ersichtlich: <https://github.com/fuerbringer/bma/blob/master/src/helper.js#L15>

```

4:   for all zeile ∈ raster do                                ▷ y-Achse
5:       for all zelle ∈ zeile do                                ▷ x-Achse
6:           if Ist zelle ein Startpunkt then
7:               startPunkt ← zelle                                ▷ startPunkt entspricht der aktuellen Zelle
8:           else if Ist zelle ein Endpunkt then
9:               endPunkt ← zelle                                ▷ endPunkt entspricht der aktuellen Zelle
10:  return startPunkt und endPunkt

```

Übergibt man PathFinding.js nun die nötigen Parameter des Rasters und dessen Start- und Endpunkte, so erhält man den Weg von Start bis Ende. Wiederholt man dies mehrere Male mit verschiedenen Pathfindern, so ermöglicht man den direkten Vergleich der Pathfinding-Algorithmen. Dies ist eine wichtige Kernfunktion dieser Berufsmaturitätsarbeit.

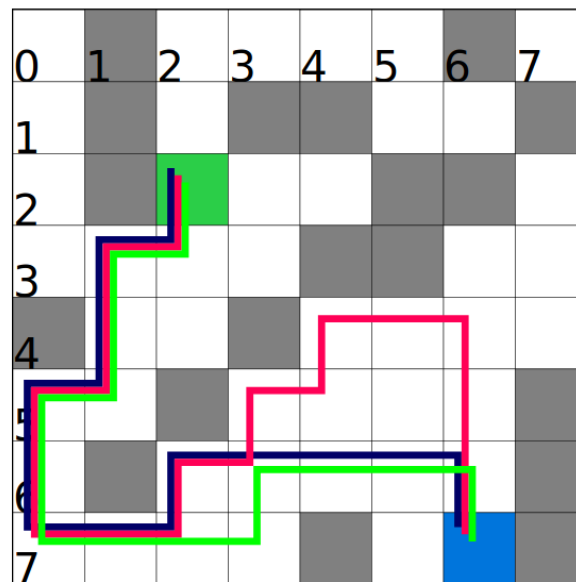


Abbildung 2.8: Ein von unserer Webapplikation generiertes 8×8 Raster mit Wänden und gelöstem Weg. Quelle: Eigenleistung

2.2.2 Auswertung und Merkmale

Zur Auswertung werden Merkmale definiert, welche bei der Statistik relevant sind. Es folgen die drei wichtigsten Merkmale beim Vergleich der Pathfinder in der Webapplikation.

Zurückgelegter Weg

Der zurückgelegte Weg ist aus praktischer Sicht gesehen ein bedeutendes Merkmal, da man natürlich möchte, dass der resultierende Weg möglichst kurz ausfällt. Die Programmbibliothek PathFinding.js liefert den Weg in einer Liste mit einer Koordinate pro Element. Daraus folgt, dass der Weg s der Anzahl der Elemente der Liste entsprechen muss, wenn man nach Path-

Finding.js Sprünge oder Luftlinien ausschliesst⁹. Es gibt demnach keine Geraden, ausser zwischen zwei Benachbarten Punkten. Diagonalen würden nach dieser Logik bei einem diagonalen Sprung als eine Weeinheit gelten.

$$s = n \begin{cases} P_1(0,0), \\ P_2(1,0), \\ \dots \\ P_n(x,y) \end{cases}$$

Operationen

Ein Algorithmus, der für das gleiche Resultat weniger Schritte (Operationen) tätigen muss, ist grundsätzlich besser als ein anderer. Zur Erfassung der Anzahl Operationen mussten die internen Routinen der Programmbibliothek PathFinding.js angepasst werden, da sie solche Messungsmerkmale nicht standardmässig liefert. Dazu wurde der Quellcode von PathFinding.js kopiert¹⁰ und in die Routinen der drei ausgewählten Pathfinder im äusseren und inneren Loop eingefügt. Folgende Dateien¹¹ wurden dafür angepasst:

- src/finders/AStarFinder.js
- src/finders/BreadthFirstFinder.js

Zu beachten ist, dass die Datei des BestFirstFinders nicht angepasst werden musste, da dieser auf dem A*-Finder basiert und somit alle Änderungen vom A*-Finder vererbt.

Rechenzeit

Ein bedeutendes Merkmal ist die Rechenzeit, also die Zeit, in der der Pathfinder seine Arbeit verrichtet hat. Um diese zu berechnen, wird unmittelbar vor der Ausführung der Wegfindungsroutine des Pathfinders der Zeitstempel t_0 der jetzigen Zeit zwischengespeichert und danach vom Pathfinder die Arbeit verrichtet. Direkt nach dem Beenden der Wegfindung wird ein zweites Mal ein Zeitstempel t_1 zwischengespeichert. Die Differenz ergibt die Zeit t in Millisekunden¹², die für die Berechnung benötigt wurde.

$$t = t_1 - t_0 \quad (2.1)$$

⁹Ein von PathFinding.js berechneter Weg: <https://pathfindingjs.readthedocs.io/en/latest/user-guide/getting-started/>

¹⁰Diesen Prozess nennt man bei freier Software "Forking".

¹¹Die genauen Anpassungen sind unter dem folgenden Link verfügbar: <https://github.com/fuerbringer/PathFinding.js/commit/51034158638ad7aaa9c5142a827bd4d3de2b8786#diff-22360b18a43ac33ae398e44b998101c7>

¹²Die Einheit Millisekunden ergibt sich aus der Schnittstelle `Performance.now()` von JavaScript

Die Werte t sind pro Pathfinder in der Webapplikation unter “Messung für diesen Einzelversuch:” unter dem Raster ersichtlich. Diese Berechnung demnach pro Durchlauf dreimal getätigt.

Aus diesen drei Merkmalen folgen zusammengefasst die in unserer Webapplikation vermessenen Werte der Pathfinding-Algorithmen.

Messung für diesen Einzelversuch:

AStarFinder:	38 Zellen, 265 Operationen, 2 ms
BestFirstFinder:	42 Zellen, 133 Operationen, 1 ms
BreadthFirstFinder:	38 Zellen, 2139 Operationen, 5 ms

Navigation: Pfeiltasten

Abbildung 2.9: Vergleicherausschnitt. Quelle: Eigenleistung

3. Anhang

3.1 Screenshots der Webapplikation

3.1.1 Pathfinder-Vergleicher

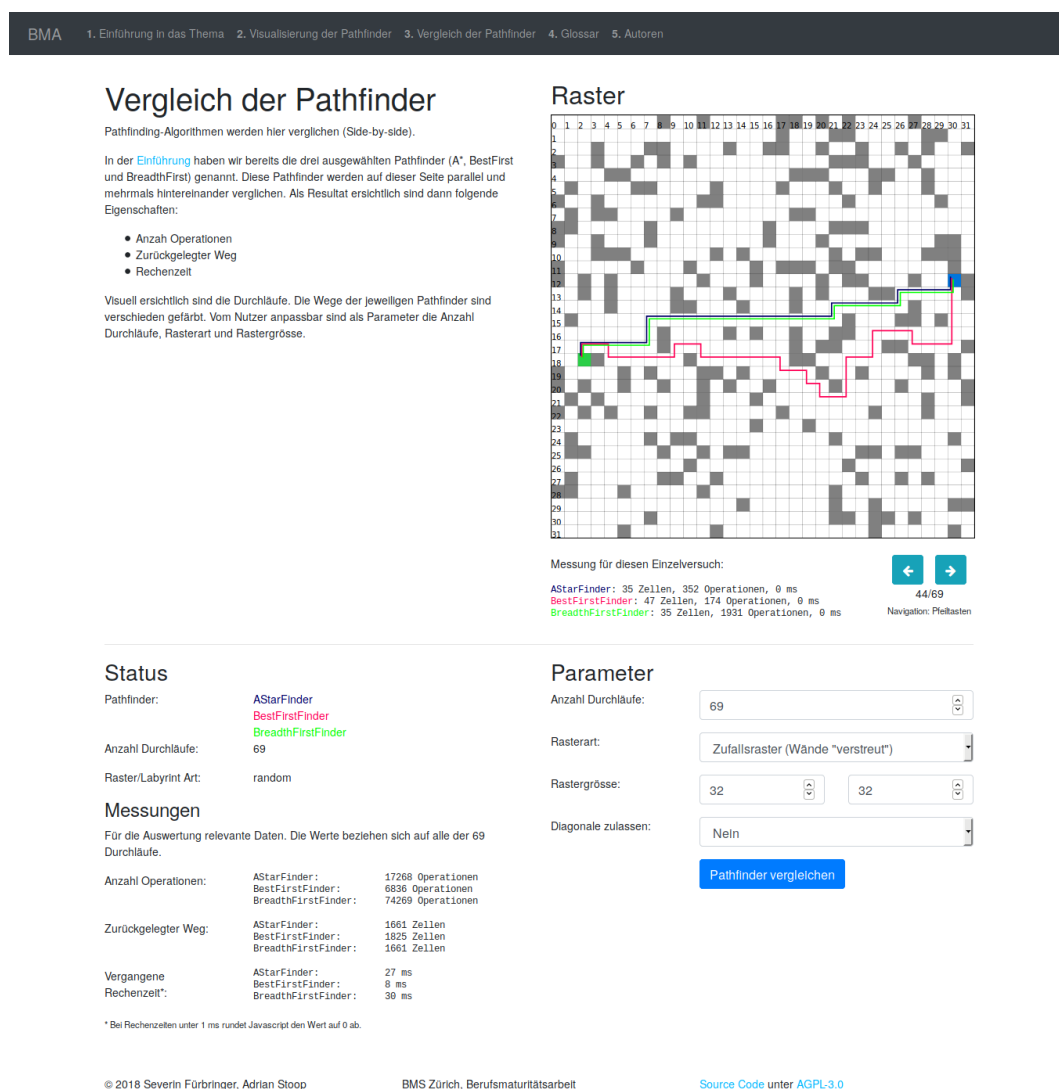


Abbildung 3.1: Pathfinder-Vergleicher-Page. Quelle: Eigenleistung

3.2 Danksagung

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

3.2.1 Schriftsetzung

Die Schriftsetzung wurde durch die freie Software \LaTeX und deren Autoren Leslie Lamport et. al. ermöglicht.

4. Glossar

Fachwörter- und Begriffsverzeichnis

In diesem Kapitel werden wichtige Wörter aus der Fachsprache erklärt, die in unserer Arbeit oftmals vorkommen.

A* Sprich “A Star”. Ein Pathfinding Algorithmus (siehe Fachbegriff Algorithmus). Dieser nutzt zusätzlich heuristische Mittel, um den Weg zu berechnen (siehe Heuristik).

Algorithmus Ein Ablauf, Prozess oder Programm, der eine Liste mit Anweisungen schrittweise befolgt, um Daten umzuwandeln.

Back-End Der Server—mit guten Systemtechnikern rund um die Uhr stellt das Back-End die Website unter einer URL, wie `bma.fuerbringer.info`, zur Verfügung.

BestFirstFinder Ein Pathfinding-Algorithmus. Nicht zu verwechseln mit dem BreadthFirstFinder.

BreadthFirstFinder Ein Pathfinding-Algorithmus. Nicht zu verwechseln mit dem BreadthFirstFinder.

Dijkstra Ein Pathfinding Algorithmus (siehe Fachbegriff Algorithmus). Dieser nutzt keine heuristische Mittel und kommt in unserer Arbeit nicht direkt vor.

Front-End Der Browser—der Teil einer Website oder Webapplikation, den der Benutzer sieht und mit dem er interagiert. In unserem Fall werden die wichtigsten Berechnungen im Front-End verrichtet.

Heuristik Hilfsalgorithmus, unter anderem für Pathfinder, der zusätzlich hilft zwischen einzelnen Schritten die Kosten für eine mögliche Operation einzuschätzen.

Matrix Anordnung von Elementen. In unserem Fall ein zweidimensionales Raster, in dem Wände, Korridore und Wege platziert sind.

Operationen Eine “Operation” beschreibt in unserem Fall einen Zyklus eines Pathfinders. Je mehr Zyklen ein Pathfinder in Relation zu einem anderen benötigt, desto weniger effizient ist er.

Parameter Die Randbedingungen für einen Durchlauf bzw. ein Experiment.

PathFinding.js Die Zentrale Programmbibliothek, deren Pathfinder Implementationen wir in unserer Webapplikation nutzen.

Pathfinder Eine Art Algorithmus, der in einem gegebenen Raum mit eingezeichnetem Start- und Endpunkt den schnellsten weg Findet.

recbacktracker “Recursive Backtracker, ein Labyrinthalgorithmus, den wir in unserer Webapplikation zur Generierung von perfekten (immer lösbaren) Labyrinthen verwenden. Dieser Algorithmus kommt im Vergleich nicht zum Zug.

Pseudocode Definiert einen Algorithmus schriftlich in einer allgemein verständlichen Sprache, damit dieser in beliebigen Programmiersprachen umgesetzt werden kann.

PHP Eine freie serverseitige Programmiersprache, die seit über zwei Jahrzehnten einen hohen Marktanteil hat.

JavaScript Nicht zu verwechseln mit Java. Eine clientseitige Programmiersprache, die in jedem Browser integriert ist. Sie wird hauptsächlich zwecks der Interaktivität benutzt.

Implementieren/Implementation Im Kontext der Software wird damit die Planung und Entwicklung des Quellcodes bezeichnet.

Pixelmatrizen Eine Liste mit Pixeldaten, die zum Beispiel bei Computergrafikapplikationen gebraucht wird. Der “Inhalt” des Bildschirm kann ganzheitlich in einer Pixelmatrix abgelegt werden.

Prozedural Beschreibt programme, die schrittartig ablaufen.

Routine Unter diesem Begriff kann man im Rahmen dieser Arbeit einen Teil eines Algorithmus, einer Funktion oder einer Prozedur verstehen.

Unix und GNU/Linux Betriebssysteme abstammend vom AT&T UNIX.

Literaturverzeichnis

- [1] Xueqiao Xu et al., *PathFinding.js*, <https://github.com/qiao/PathFinding.js>, Quellcode auf GitHub, Letzter Aufruf: 28. Januar 2019
- [2] Xiao, Cui; Hao Shi, *A*-based Pathfinding in Modern Computer Games*, IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, Januar 2011.
- [3] Fürbringer, Severin; Stoop, Adrian, *BMA Quellcode*, <https://github.com/fuerbringer/bma>, 28. Januar 2019
- [4] Fürbringer, Severin; Stoop, Adrian, *BMA-Webapplikation*, <https://bma.fuerbringer.info>, 28. Januar 2019

Abbildungsverzeichnis

2.1	Struktur der Webapplikation.	4
2.2	Konzept der Einführungsseite.	5
2.3	Benutzeroberflächenkonzept des Pathfinding-Visualisierers.	6
2.4	Benutzeroberflächenkonzept des Pathfinder-Vergleichers.	7
2.5	Ein von unserer Webapplikation generiertes 8×8 Raster.	9
2.6	Raster mit Frequenzparametern $freq = 2$, $freq = 4$ und $freq = 16$ im Wandge- nerator.	10
2.7	Ein von unserer Webapplikation generiertes 8×8 Raster mit Wänden und mar- kierten Start- und Endpunkten.	11
2.8	Ein von unserer Webapplikation generiertes 8×8 Raster mit Wänden und gelöstem Weg.	12
2.9	Ausschnitt aus dem Pathfinding-Vergleicher für einzelne Vergleiche.	14
3.1	Ein vollständiger Screenshot des Pathfinder-Vergleichers.	15