Software Engineering Projekt „Roboter“

**Ihr**

**Firmenlogo**

**hier**

Seminararbeit

erstellt im Rahmen der Veranstaltung:

Projektmanagement & Softwareengineering

Vorgelegt von: Jens Grehling, Thomas Häusser, Simon Weyand,

Maximilian Hasenstab, Jan-Jörn Schulz und Tobias Feller

Studiengruppe: WS-13-II

Matrikelnummer: tba

Anzahl der Wörter: [xxxxx]

Akademischer Gutachter: Gamer Michael

Abgabedatum: 26.06.2015

Inhalt

[1 Problembeschreibung 3](#_Toc420862144)

[2 Darstellung von Lösungsalternativen 3](#_Toc420862145)

[3 Begründete Auswahl einer Alternativen 3](#_Toc420862146)

[4 Beschreibung des Algorithmus zur Lösung 3](#_Toc420862147)

[4.1 Implementation des Algorithmus 3](#_Toc420862148)

[4.1.1 Programmiersprache / Entwicklungsumgebung 3](#_Toc420862149)

[4.1.2 Aufbau des Programms 4](#_Toc420862150)

[4.1.3 Datenstrukturen 4](#_Toc420862151)

[5 Dokumentierter Programmcode 4](#_Toc420862152)

[6 Beispielausgabe des Programms 4](#_Toc420862153)

# Problembeschreibung

**Aufgabenstellung:**

Ein Roboter

„Gegeben ist eine, theoretisch unendlich große, zweidimensionale, Ebene, welches in

quadratischen Felder eingeteilt ist. Auf einem Feld steht ein Roboter. Dieser hat eine

Bewegungsrichtung. Das Verhalten des Systems ist durch folgende beiden Regeln

beschrieben:

1. Falls das Feld unter dem Fokus weiß ist, so ändert dieser die Bewegungsrichtung

im 90\_ nach rechts, invertiert die Feldfarbe und bewegt sich ein Feld in Richtung

der (neuen) Bewegungsrichtung.

2. Falls das Feld unter dem Fokus schwarz ist, so ändert dieser die Bewegungsrichtung

im 90\_ nach links, invertiert die Feldfarbe und bewegt sich wiederum ein

Feld in Richtung der (neuen) Bewegungsrichtung.

Entwerfen und implementieren Sie einen Algorithmus, der das Verhalten des Roboters

simuliert und untersuchen Sie das Verhalten dieses dynamischen Systems!“

Ausgangspunkt dieses Projekt ist die vermerkte Aufgabenstellung.

# Darstellung von Lösungsalternativen

# Begründete Auswahl einer Alternativen

# Beschreibung des Algorithmus zur Lösung

## Implementation des Algorithmus

### Programmiersprache / Entwicklungsumgebung

Das Programm wurde mithilfe der Programmiersprache JavaScript realisiert. Grund für die Auswahl von Java-Script ist die hohe Flexibilität von JavaScript so wie die weitreichende Plattformunabhängigkeit.

### Aufbau des Programms

Das Programm setzt sich aus den folgenden Dateien zusammen:

|  |  |
| --- | --- |
| **Inhalt** | **Datei** |
| **Es werden für Anwendung benötigte Frameworks und Bibliotheken eingebunden und die Benutzeroberfläche erzeugt. Außerdem wird der Programmcode der Anwendung geladen.** | Index.html |
| **Hier wird die Darstellung der für das Programm benötigten Objekte definiert.** | Styles.css |
| **Grafik zur Darstellung der Blickrichtung des Roboters.** | Arrow-top.png |
| **Hier befindet sich der Programmcode der Anwendung.** | App.js |
| **Einbinden der Javascript-Bibliothek jquery.** | jquery-1.11.0.min |
| **Einbinden des Materialize-Frameworks.** | /materialize/css/materialize.css |

### Verwendete Datenstrukturen

Da es sich bei Javascript um eine klassenlose Programmiersprache handelt, werden die verwendeten Variablen und ihre Funktion im Folgenden erläutert.

|  |  |
| --- | --- |
| **Funktion** | **Variable** |
| **Definiert die Standardfarbe des Spielfelds.** | Var defaultColour |
| **Zählt den aktuellen Iterationsschritt.** | Var counter |
| **Nimmt die vom Anwender eingestellten Iterationsschritte entgegen.** | Var counter\_s |
| **Definiert den Zeitwert bis zur Ausführung des nächsten Iterationschrittes (Anwendereingabe).** | Var timeout |
| **Hier werden die bereits besuchten Felder für Auswertungen gespeichert.** | Var fieldsVisited |
| **Die Größe des Spielfelds wird gespeichert, befindet sich der nächste Iterationsschritt außerhalb des Spielfeldes so wird dieses vergrößert.** | Var area\_length |
| **Aktuelle Iteration für die Ausgabe auf der HTML-Darstellung.** | Var current\_iteration |
| **Variable speichert ob sich die Anwendung aktuell im pausierten Zustand befindet (pause=1)** | Var paused |
| **Speichert die aktuellen Parameter der Anwendung bei Ihrer Pausierung.** | Var savedVarsonPause |

Tabelle 4‑1 : Verwendete Datenstrukturen

### Verwendete Funktionen

Im Folgenden sollen die Funktionen der Anwendung erläutert werden. Um das Verständnis der Anwendung zu vereinfachen wird hierbei, soweit dies möglich chronologisch nach dem Aufruf der Funktionen innerhalb des Programms vorgegangen.

**function start**

Die ***function start()*** ruft die ***function*** ***setPause()*** auf und übergibtdas Parameter 0. Weiterhin wird die ***function intialize*** aufgerufen.

Desweiteren werden zusätzliche Menüpunkte freigeschaltet zum einen die Möglichkeit den Algorithmus zu pausieren, sowie die Anzeige der Iterationschritte und die Anzeige der Blickrichtung des Roboters.

**function setPause**

Diese Funktion erwartet den Übergabeparameter 0 (Pausieren) oder 1 (Fortsetzen) und steuert hiermit die Ausgabe des entsprechenden Buttons der Anwendung.

**function initialize**

In dieser Funktion werden zunächst die Startparameter intialisiert. Hierfür werden den im Abschnitt Verwendete Datenstrukturen beschriebenen Variablen die folgenden Parameter übergeben.

|  |  |
| --- | --- |
| **Variablenname** | **Initialisierungsparameter** |
| **defaultColour** | 0 (Starten mit Ausgangsfarbe weiß) |
| **counter** | 1 (da es bei geringeren Werten keine Ausgabe gibt) |
| **fieldsvisited** | [] (leeres Array welches mit dem Aufruf der function nextStep() gefüllt wird) |
| **area\_length** | 0 (zu Beginn 0 da sich diese erst mit Ablauf des Algorithmus aufbaut) |
| **current\_iteration** | 1 (siehe counter) |
| **paused** | 0 (bei 1 wird die function nextStep() ausgeführt) |
| **savedVarsonPause** | [](wie im Abschnitt Verwendete Datenstrukturen beschrieben wird dieser Array erst beim Pausieren der Anwendung gefüllt) |

Tabelle 4‑2 : function intialize

Im nächsten Schritt werden die vom Nutzer eingegebenen Variablen timeout (Iterationsgeschwindigkeit) und counter\_s (Anzahl Iterationen) entgegen genommen

Es wird nun das Spielfeld aufgebaut. Dieses ist wie bereits erwähnt als Array erstellt. Um die Einfärbung des Spielsfelds zu bestimmen wird die ***function getColor()*** aufgerufen.

Im Anschluss wird die Blickrichtung des Roboters ausgelesen und in der Variable ***orientation*** gespeichert. Der Startpunkt des Roboters ist in der Variable ***position*** mit dem Feld [1,1] festgelegt.

Die ***function*** endet mit dem Aufruf der ***function nextStep***.

**function getColor()**

Die ***function getColor()*** prüft zunächst den Parameter ***defaultcolour*** und färbt in den Fällen das der Wert black(1) oder white(0) ist entsprechend ein. Falls als Feldfarbe random gewählt wurde, werden Zufallszahlen generiert jede Zufallszahl größer als 0,5 ergibt ein schwarzes Feld jede Zufallszahl kleiner als 0,5 ein weißes Feld.

***function nextStep***

## Testen der Anwendung

Da es sich bei der zu testenden Anwendungen um einen eher simplen Algorithmus handelt sollen die Tests der Anwendung händisch erfolgen. Zu Testen ist zunächst die Korrektheit der Algorithmus. Weiterhin sollen mögliche Bedienfehler durch den Anwender ausgeschlossen und die Plattformunabhängigkeit geprüft werden.

### Korrektheit des Algorithmus

Hierfür soll ein Beispiel gewählt werden an dem die Korrektheit nachvollzogen werden kann. Es sollten alle Fälle die im Verlauf des Programmablaufs auftreten können behandelt werden. Die folgenden Fälle können eintreten:

* Der Roboter steht auf einem weißen Feld.
* Der Roboter steht auf einem schwarzen Feld.
* Der Roboter überschreitet die bisher vorhandenen Spielfeldgrenzen.
* Das eingestellte Limit an Iterationen ist erreicht.

### Bedienfehler durch den Anwender

Bei der Nutzung der „Ein Roboter“-Anwendung hat der Nutzer verschiedene Interaktionsmöglichkeiten mit dem Programm. Beim Start kann er zunächst wählen ob er das ***Programm*** nutzen oder die ***Erklärung*** lesen möchte. Ein Umschalten zwischen diesen beiden Menüpunkten ist problemlos möglich. Der Menüpunkt ***Erklärung*** wird im Folgenden nicht weiter berücksichtigt, da der Anwender keine weiteren Interaktionsmöglichkeiten hat.

Im Menüpunktprogramm kann der Nutzer dem Roboter unterschiedliche Startparameter mitgeben die folgende Tabelle soll eine Übersicht darüber geben, welche eingeben für den Anwender nicht möglich sein sollen:

|  |  |
| --- | --- |
| **Eingabe** | **Kritischer Wertebereich** |
| **Timeout in ms** | Werte < 0 |
| **Iterationen** | Werte < 1 (da keine Ausgabe erfolgt) |
| **Ausrichtung** | Nicht zutreffend(keine Eingabe außer Vorgabe möglich) |
| **Standardfarbe** | Nicht zutreffend(keine Eingabe außer Vorgabe möglich) |
| **Start** | Nicht zutreffend(nur eine Aktion möglich) |
| **Pausieren** | Nicht zutreffend(Button ist erst bei Start des Algorithmus aktiv) |

Tabelle 4‑3 : Bedienfehler

## Plattformunabhängigkeit

Neben den Entwicklungssystemen (Windows 7 64 bit) soll die Anwendung auch auf mobilen Endgeräte (Android) und anderen Desktop-Betriebssystemen (Mac OSX und Linux getestet werden).

# Dokumentierter Programmcode

# Beispielausgabe des Programms