|  |  |  |
| --- | --- | --- |
| Sokoban Solver BFH | HS 2016  HS 2016 | |
| Dieses Dokument fasst die Arbeit an unserem Java-Projekt zusammen. | | [Geben Sie den Untertitel des Dokuments ein] |

Pascal Ammon ammop2

Gabriel Wyss wyssg2

Inhalt

[Projektumfeld 2](#_Toc440809899)

[Personen / Stakeholders 2](#_Toc440809900)

[Hardware 2](#_Toc440809901)

[Rahmenvorgaben 2](#_Toc440809902)

[Abgrenzung 2](#_Toc440809903)

[Initalisierung 3](#_Toc440809904)

[Projektziele 3](#_Toc440809905)

[Funktionelle Aspekte 3](#_Toc440809906)

[Karten speichern 3](#_Toc440809907)

[Karten lösen (Solver) 3](#_Toc440809908)

[Sokoban spielen 3](#_Toc440809909)

[Spieler schnell verschieben 3](#_Toc440809910)

[Karte erstellen/editieren 4](#_Toc440809911)

[Solver Performance 4](#_Toc440809912)

[Anwendungsstabilität 4](#_Toc440809913)

[Projektverlauf 5](#_Toc440809914)

[Bemerkungen zum Verlauf 6](#_Toc440809915)

[SokobanSolver 7](#_Toc440809916)

[Lösungsvorschläge 7](#_Toc440809917)

[Lösung 1: 2-Dimensionales Array 7](#_Toc440809918)

[Lösung 2: Felder enthalten Pointer auf ihre Nachbarfeldern 7](#_Toc440809919)

[Unser Sokoban Solver 8](#_Toc440809920)

[Verbesserungsmöglichkeiten Solver 8](#_Toc440809921)

[Flussdiagramm Solver 9](#_Toc440809922)

[Struktur 10](#_Toc440809923)

[Logikteil 10](#_Toc440809924)

[GUI 11](#_Toc440809925)

[What next? 12](#_Toc440809926)

[Bedienung 12](#_Toc440809927)

[Spielmodus 12](#_Toc440809928)

[Editormodus 12](#_Toc440809929)

[Reverse Play Modus 12](#_Toc440809930)

[Glossar 13](#_Toc440809931)

# Projektumfeld

## Personen / Stakeholders

|  |  |  |
| --- | --- | --- |
| Name | Rolle | Email |
| Peter Schwab | Stakeholder | [peter.schwab@bfh.ch](mailto:peter.schwab@bfh.ch) |
| Pascal Ammon | Auftragnehmer | [pascal.ammon@gmail.com](mailto:pascal.ammon@gmail.com) |
| Gabriel Wyss | Auftragnehmer | [gabriel.wyss@gmail.com](mailto:gabriel.wyss@gmail.com) |

## Hardware

Das Produkt ist für folgende Hardware bestimmt:

|  |  |
| --- | --- |
| Version | Hardware |
| Desktop | Intel basiert  Windows 7 oder neuer |

## Rahmenvorgaben

Die Rahmenvorgaben sind durch das Modul Projekt1 gegeben.

## Abgrenzung

Dieses Projekt umfasst die Entwicklung der Java-Applikation für Desktop. Die Hardware, Betriebssystem oder andere Software gehören nicht zu diesem Projekt.

Das Produkt hat Schnittstellen mit dem Benutzer und dem darunter liegenden Betriebssystem. Es sind keine Schnittstellen mit weiteren Software-Systemen vorgesehen.

Sokoban-Projekt

Betriebssystem:

Windows oder Linux/MAC

Hardware

Java Runtime Environment

Sokoban Prototyp

# Initalisierung

## Projektziele

Ziel des Projekts ist die Erstellung eines Sokoban-Spiels (gemäss Wikipedia <https://en.wikipedia.org/wiki/Sokoban> ). Zu diesem Zweck soll mit Java ein Prototyp entwickelt werden.

Ebenfalls soll in diesem Projekt die Funktion und Performance eines selbst erstellten, automatischen Lösungsalgorithmus aufgezeigt werden.

Es sollen weitere Features implementiert werden, welche im Rahmen dieses Projekts genauer definiert werden. Das Endprodukt besteht aus einem Prototypen der Anwendung inklusive Dokumentation.

## Funktionelle Aspekte

### Karten speichern

Die Anwendung kann Spielkarten auf dem Dateisystem abspeichern und einlesen. Das im Internet gängige Textformat der Karten soll dazu benutzt werden.

|  |  |
| --- | --- |
| Dateiformat | ASCII |
| Symbol für Key | . |
| Symbol für Player | @ |
| Symbol für Target | X |
| Symbol für Ground | Y |
| Symbol für Blank | B |

Gemäss: <http://www.sourcecode.se/sokoban/levels>

### Karten lösen (Solver)

Die Applikation löst das Spiel über einen noch zu definierenden Algorithmus und zeigt dem Benutzer die Lösung dar. Dazu kann der Benutzer im Spiel die Lösungsfunktion benutzen.

### Sokoban spielen

Das Sokoban-Spiels kann gemäss den bekannten Spielregeln auf einer grafisch ansprechenden Oberfläche mittels Maus und Tastatur gespielt werden.

Der Benutzer kann seinen Avatar/Spielfigur mit den 4 WASD-Tasten nach links, rechts, rauf und runter navigieren. Der Avatar kann nur auf den Feldern bewegt werden, die gemäss Spielregeln dafür vorgesehen sind.

Der Benutzer kann zwischen mehreren grafischen Styles („Themes“) auswählen.

### Spieler schnell verschieben

Mittels eines Klicks auf die Karte kann der User die Spielfigur augenblicklich auf das betroffene Spielfeld verschieben. Die Aktion wird nur ausgeführt, wenn die mit einem erlaubten Spielzug möglich ist. Es darf zum Beispiel keine Mauer im Weg stehen. Es muss ein Breitensuch-Algorithmus eingesetzt werden.

### Karte erstellen/editieren

Der Benutzer kann eigene Karten für das Spiel bauen oder bestehende Karten editieren. Der Editor erlaubt es, eine Kartengrösse (in Spielsteinen), Kartennamen und Kartenschwierigkeit anzugeben Während dem Bau stellt die Applikation die Lösbarkeit sicher. Es dürfen nur lösbare Karten gebaut werden. Beim Bauen der Karte werden automatisch nur lösbare Bauoptionen angezeigt. Die Karte kann im Karteneditor nach Schwierigkeitsgrad klassifiziert werden. Das Endprodukt wird mit mindestens 3 Beispielkarten geliefert.

### Solver Performance

Durch Nutzung von im Internet frei verfügbaren und selbst programmierten Solver-Lösungen soll der optimale Solver-Algorithmus für Sokoban gefunden werden. Es sind auch Optimierungen von bestehenden Algorithmen denkbar.

Der Solver des Prototyps löst Spielkarten mit 50 mal 50 Spielfeldern in unter drei Minuten.

### Anwendungsstabilität

Die Anwendung überprüft Benutzereingaben auf Korrektheit, um Anwendungsstabilität zu gewährleisten. Ebenfalls werden die möglichen Exceptions abgefangen, damit der Prototyp stabil wird.

Die Stabilität wird folgendermassen getestet und bewertet:

* Übersteht Problem mit Filesystem (zB. Keine Schreibrechte).
* Übersteht und/oder filtert Eingabe von allen Sonderzeichen im UTF-8 Zeichensatz

# Projektverlauf

|  |  |  |  |
| --- | --- | --- | --- |
| Wann | Dauer in h | Was | Wer |
| 18.09.2015 | 6 | Playground Project (Initalisierung)  Was wollen wir machen?  Klassen identifizieren | Pascal |
| 20.09.2015 | 3 | Playground Project | Pascal |
| 21.09.2015 | 8 | Playground Project  Menu erstellt  Maps von Text-File einlesen  Sokoban spielen (Hauptfunktion) | Gabriel |
| 24.09.2015 | 1 | Github Projekt aufsetzen (Source Code Management) | Pascal |
| 23.09.2015 | 1 | Erste Projektsitzung mit Herr Schwab  Themen: Welche Ziele? Welche Technologien? | Alle |
| 25.09.2015 | 4 | Konkrete Requirements identifizieren | Pascal  Gabriel |
| 02.10.2015 | 6 | Klassendiagramm erstellen  Arbeitsjournal erstellen | Pascal  Gabriel |
| 04.10.2015 | 2 | Zweite Projektsitzung mit Herr Schwab  Themen: Solver-Ansätze | Alle |
| 06.10.2015 | 4 | Klassenstruktur im IDE erstellen  Verwendung eingelesener Maps im Play-Teil | Pascal |
| 07.10.2015 | 8 | Dokumentation | Gabriel |
| 18.11.2015 | 2 | Dritte Projektsitzung mit Herr Schwab  Themen: Weitere Features, Solver | Alle |
| 19.11.2015 | 10 | GUI neu aufsetzen mit JavaFX  Karten erstellen (MAP Editor)  Karten speichern | Gabriel  Pascal |
| 23.11.2015 | 10 | Spieler springen lassen im Spielmodus  Klassendesign überarbeiten  Spielerlogin für das Anzeigen von Spielresultaten  Analyse fremde Solver-Code | Gabriel  Pascal |
| 26.11.2015 | 10 | Analyse und Integration fremde Solver  GUI | Pascal  Gabriel |
| 02.12.2015 | 2 | Vierte Projektsitzung mit Herr Schwab  Themen: Abgabe, Solver | Alle |
| 9.12.2015 | 4 | Integration Fremde Solver  GUI | Pascal  Gabriel |
| 17.12.2015 | 10 | Bau unserer eigenen Solver-Lösung | Pascal  Gabriel |
| 22.12.2015 | 10 | Bau unserer eigenen Solver-Lösung | Pascal |
| 17.01.2016 | 10 | Testing  Debugging | Pascal  Gabriel |
| 18.01.2016 | 10 | Dokumentation  Bereit für Abgabe | Pascal  Gabriel |

## Bemerkungen zum Verlauf

Generell haben wir uns im ersten Teil des Projekts mit GUI- und funktionellen Aspekten befasst. Wir haben sogar freien Sokoban-Solver-Code aus dem Internet in unser Projekt eingebunden und unser Projekt war eigentlich schon fertig. Da wir noch Zeit hatten, haben wir gegen das Ende des Projekts noch unseren eigenen Solver gebaut (gemäss Vorschlägen von Sokobano.de). Die Dokumentation wurde auch am Ende des Projekts erstellt. Ebenfalls haben wir uns zuerst für Java Swing als GUI-Framework entschieden. Da es für Spiele sehr ungeeignet ist, haben wir das GUI später komplett neu geschrieben mit JavaFX, welches zwar immer noch keine Animationen ermöglicht, aber deutlich besser aussieht.

Technische Probleme hatten wir während des Projekts sehr wenig. Einzig die Dateipfäde haben uns am Anfang Probleme bereitet, welche eine Ausführung auf Linux verhindert haben. Es gibt aber eine Notation für Pfäde, welche sowohl unter Linux als auch unter Windows funktioniert.

# SokobanSolver

Da es sich hier um den Kernteil unseres Projekts handelt, haben wir hier am meisten Zeit investiert.

## Lösungsvorschläge

### Lösung 1: 2-Dimensionales Array

Das Spielfeld wird in einem 2D Array abgebildet, die 1Dimension repräsentiert die Zeilen des Spielfelds, die 2Dimension die Spalten. Wenn der Player durch das Feld navigiert werden soll müssen nur die Position vertauscht werden. Nachteil ist das nur ein Rechteckiges Spielfeld denkbar ist.

### Lösung 2: Felder enthalten Pointer auf ihre Nachbarfeldern

Auf jedem Feld werden Zeiger auf die Nachbarfelder gespeichert. Beim Navigieren werden diese Zeiger überschrieben. Vorteil das Spielfeld ist an keine Form gebunden.

* Lösung 2 ist einfacher zu bewirtschaften. Lösung 2 ist nicht 2D.

## Unser Sokoban Solver

Die Idee ist einen Solver zubauen welcher in kurzer Zeit mittelschwere Sokoban-Probleme lösen kann.

**FeldTypen für den Solver**

|  |  |  |
| --- | --- | --- |
| Name | Character in Solver | Character in abgespeicherten Mapfiles |
| Border | B | X |
| Target | T | . |
| Key | K | \* |
| Field | F | **' '** || @ |
| FieldOutside | X |  |
| DeadLockField | D |  |

Die Map wird in ein zweidimensionales Array geladen.

* Da sich der Spieler frei bewegen kann wird dieser als Field gemapped. Die freien Felder ausserhalb vom Rand werden nicht geladen.
* DeadLockField erkennen
* Kollision mit Wand? An Wand entlang gehen. Wenn Feld in einer Ecke 🡪 Deadlock Field.

Jeder Spielzug „Move“ wird in einer HashTable gespeichert. Als Key wird der Hash vom Pfad verwendet. So kann mit Hilfe von

* HashTable.Get(„Hash from Path“)

aufgezeigt werden ob der Pfad schon vorhanden ist.

Wann ist ein Zug erlaubt?

1. Ziel ist kein Border
2. Ziel ist kein Deadlock Field

* Strecke Zum Ziel berechnen:

Ergibt sich in Anzahl Zügen nach rechts/links und Zügen nach oben/unten.

### Verbesserungsmöglichkeiten Solver

Wir haben bemerkt, dass unser Solver manchmal beim Start unnötige Züge macht. Man müsste eine weitere Prüfung machen, um die Routen nach der Erstellung erneut zu prüfen.

### Flussdiagramm Solver



# Struktur

Wir haben unsere Software in folgende Klassen aufgeteilt:

## Logikteil

|  |  |
| --- | --- |
| Klassenname | Zweck |
| ChangeItem | Kann ein Feld auf der Karte verändern |
| Coordinate | Speichert eine Koordinate |
| Difficulty | Enumeration für die Schwierigkeit |
| Direction | Enumeration für die Richtungen |
| EditorHandler | Logik des Karteneditors |
| FieldList | Kartenlogik |
| FieldTyp | Enumeration der Feldtypen |
| Handler | Handle-Klasse für Main-Programm |
| Map | Speichert Karte |
| MapReader | Kann Karte von Textfile einlesen |
| MapSaver | Kann Karte in Textfile speichern |
| Mode | Enumeration für die verschidenen Programmmodi |
| Player | Speichert das Feld, welches vom Spieler gesteuert wird |
| Result | Speichert das Resultat (Anzahl Züge), das ein Spieler erreicht hat |
| ReverseHandler | Handle-Klasse für das Rückwärts-Spielen |
| ThemeReader | Kann ein Theme (Skin) für die Anwendung einlesen |
| User | Speichert einen Benutzer |
| UserReader | Kann einen Benutzer vom Texfile einlesen |
| UserSaver | Kann einen Benutzer in das Textfile speichern |

## GUI

|  |  |
| --- | --- |
| Klassenname | Zweck |
| Editor | Die Editor-Klasse |
| Login | Dialogfenster, wo sich der Benutzer einloggen kann |
| Loose | Anzeige, wenn ein Spiel verloren ist. |
| Main | Hauptframe mit Menustruktur |
| MapNew | Dialogfenster zum erstellen neuer Maps |
| MapSelect | Dialogefenster zum Wählen, welche Map gespielt werden soll |
| Playground | Das Sokoban-Spiel |
| Solver | Ansicht, wo Karten mit dem Solver gelöst werden können |
| Win | Anzeige, wenn ein Spiel gewonnen wurde. |

# What next?

Was würden wir weiter in unser Projekt einbauen, wenn wir mehr Zeit gehabt hätten?

* Anpassung der Software, damit sie auf Android laufen könnte
* Verbesserungen Solver

# Bedienung

Nachfolgend die wichtigsten Bemerkungen zur Bedienung unseres Prototyps.

## Spielmodus

|  |  |
| --- | --- |
| Aktion | Eingabe |
| Spieler bewegen | [W][A][S][D]-Tasten |
| Aktion rückgängig machen | [Ctrl]+[Z] |
| Spieler falls möglich direkt auf Feld springen | Mausklick auf entsprechendes Feld |

## Editormodus

|  |  |
| --- | --- |
| Aktion | Eingabe |
| Umschalten zwischen den einzelnen Spielsteinen zum Bauen | Klick auf Icon auf der rechten Seite |
| Spielstein bauen | Klick auf Feld auf der Karte |
| Aktion rückgängig machen | Klick auf Icon auf der rechten Seite |
| Karte speichern | Klick auf Diskettensymbol auf der rechten Seite |

## Reverse Play Modus

|  |  |
| --- | --- |
| Aktion | Eingabe |
| Spieler bewegen | [W][A][S][D]-Tasten |
| Einschalten oder ausschalten, dass der Spieler Keys bewegen kann | Klick auf Icon auf der rechten Seite |
| Karte speichern | Klick auf Diskettensymbol auf der rechten Seite |

# Glossar

|  |  |
| --- | --- |
| Wort | Beschreibung |
| Key | Das Spielfeld, das zum Gewinnen durch den Spieler auf das Target geschoben werden muss. |
| Ground | Das Spielfeld, das den „Boden“ des Spiels darstellt. |
| Target | Das Spielfeld |
| Blank | Das nicht definierte Spielfeld (für programminterne Zwecke) |
| Player | Das Spielfeld, das die Spielfigur darstellt. |
| User | Eine Person, die das Sokoban Spiel spielt. |
| Exception | Ein von der JVM unerwartetes Ereignis, welches das weiterlaufen des Programmcode verhindert. |
| JVM | Java Virtual Machine: Software, die das Ausführen von Java-Code ermöglicht. |
| Theme (oder auch Skin) | Vordefiniertes Designschema einer Applikation, das vom User geändert (ausgewählt) werden kann. |
| UTF-8 | Gängiger Zeichen-/Symbolsatz für moderne Computersysteme. |