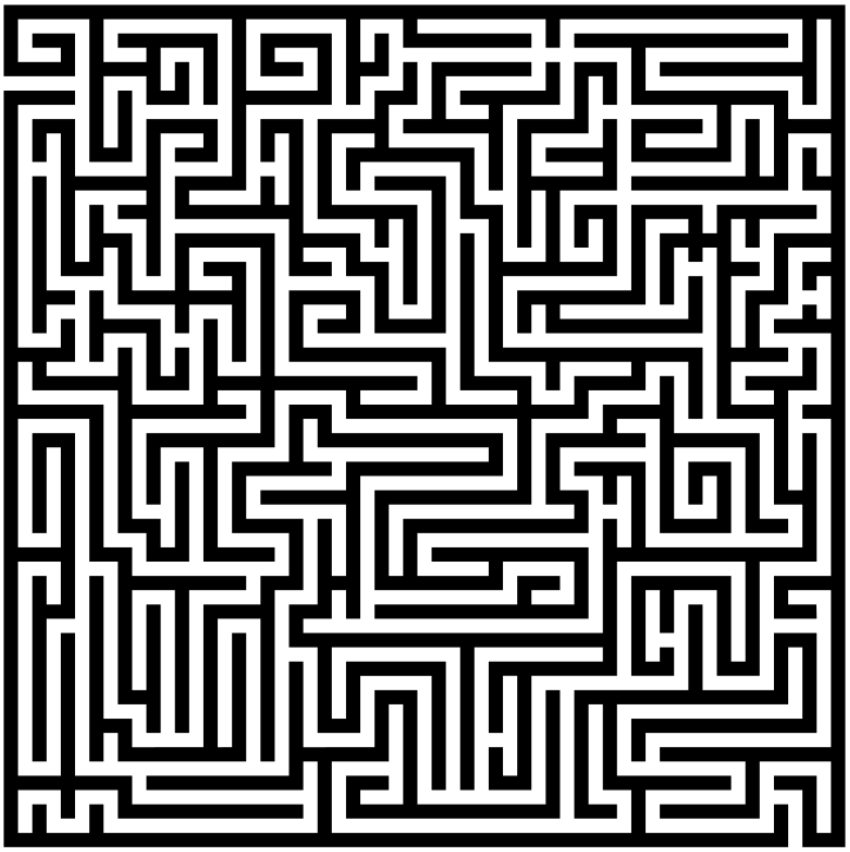
Labyrinth

Wo ist der Ausweg?



Eine Semesterarbeit an der ZHAW School of Engineering
im 2. Studienjahr von

Benjamin Bütikofer
buetikbe@students.zhaw.ch

und

Dominic Schlegel
schledom@studends.zhaw.ch

May 2013

Inhaltsverzeichnis

Einleitung

Wer als Kind schon mal in einem Malbuch den Ausweg aus einem Labyrinth gesucht hat weiss, dass dies nicht immer ganz einfach ist. Oft kommt man in eine Sackgasse und muss zurück zur letzten Kreuzung, um dort dann einen anderen Weg einschlagen zu können. Je mehr Sackgassen ein solches Labyrinth jedoch hat, desto mehr muss man sich merken. Bei mehr als fünf gemerkten Sackgassen weiss man dann plötzlich nicht mehr wo man schon einmal war und wo noch nicht.

Bei solch einem Problem kann ein Computerprogramm natürlich Abhilfe schaffen, denn heutzutage ist die Speicherkapazität fast unbegrenzt. In diesem Software-Projekt wollen wir uns deswegen vertieft mit Fragen rund um ein Labyrinth befassen.

Aufgabenstellung

Die Aufgabe dieses Projektes besteht darin, eine Applikation zu erstellen, welche ein Labyrinth generieren und anschliessend die „beste“ Lösung finden kann. Dabei ist zu beachten, dass das generierte Labyrinth einen Eingang und einen Ausgang hat. Zudem sollte es mehrere mögliche Lösungswege geben.

Ziel

- Generieren eines Labyrinths
- Den bestmöglichsten Weg mittels eines Algorithmus aus einem Labyrinth finden
- Das Verstehen, Nachvollziehen und Anwenden der Algorithmen, welche für das dynamische generieren oder das Lösen eines Labyrinths verantwortlich sind

Labyrinth Allgemein

Definition

Arten von Labyrinthen

Theorie Algorithmen

Depth First Search

Wall-Follower

A-Star

Implementation

Wahl des Umfeldes

Pattern

Testing

asdfasdfasdfasdfsdfasdf

Planung

Wir haben unser Projekt in drei Iterationen und eine Pufferphase unterteilt. Wir wollen uns als erstes um die Algorithmen kümmern, welche ein Labyrinth generieren und dazu das GUI erstellen. In der zweiten Phase des Projekts widmen wir uns der Realisierung eines Algorithmus zum Lösen eines Labyrinths und implementieren weitere Erstellungsalgorithmen. Im dritten Teil werden wir weitere Lösungsalgorithmen implementieren und das GUI vervollständigen. Natürlich halten wir uns einen gewissen Puffer für das Projektende auf. Dieser kann für Code-Reviews, Bug fixing, Dokumentation und andere Verbesserungen verwendet werden.

Die Iterationsplanung erfolgt mittels der Webapplikation Scrumdo welche unter www.scrumdo.com erreichbar ist.

1. Iteration (06.03.2013 – 27.03.2013)

Zu Begin unseres Projektes waren wir voller Elan und starteten nach der Planung direkt mit der Implementation der Tasks. Als Datenstruktur für ein Labyrinth wurde ein zweidimensionales Array aus Integern gewählt. Das GUI wurde zu diesem Zeitpunkt mit den Basiselementen aus Java gebaut.

GUI Skelet (5 Stunden geschätzt)

- Ein erstes Layout für das Anzeigen der Labyrinthe existiert (Ben)
- Der Benutzer soll im GUI auswählen können, welche Algorithmen verwendet werden (Ben)
- Die Aktionen auf den Buttons sollen implementiert werden, so dass man mittels dem GUI alles steuern kann (Ben)

Erster Labyrinth Erstellungsalgorithmus implementieren (8 Stunden geschätzt)

- Der Depth First Search Algorithmus wird implementiert (Dominic)
- Eine Basis Struktur für das erstellen von Labyrinths wird erstellt (Dominic)

Zeichnen des Labyrinths im GUI Step-by-Step (8 Stunden geschätzt)

- Labyrinth Punkt für Punkt zeichnen (Ben)

2. Iteration (03.04.2013 – 24.04.2013)

In der zweiten Iteration konnte nicht so viel an dem Projekt gearbeitet werden wie wir geplant hatten. Aufgrund von beruflichen Veränderungen bei Ben und viel Wochenendarbeit bei Dominic konnte die Iteration nicht planmässig beendet werden. Die verbleibenden Tasks wurden aber dem geänderten Zeitplan angepasst.

Während der Implementation des Prim Algorithmus und des Righ-Hand Solvers haben wir bemerkt, dass die Grundstruktur unserer Applikation mit dem 2 dimensionalen Array nicht genügt. Aus diesem Grund haben wir uns für ein Refactoring der Applikation entschieden. Wir setzten dabei nun auf einen objektorientierten Ansatz mit einer HashMap.

Dieses Refactoring hat jedoch viel Zeit in Anspruch genommen, so dass wir mit den eigentlichen Implementationen der Algorithmen etwas zurück liegen. Dank guter Planung haben wir aber einen Projektpuffer, der uns jetzt zu gute kommt.

An dieser Stelle des Projekts haben wir entschieden, dass wir den Prim Algorithmus für das Erstellen eines Labyrinths nicht implementieren, da für uns die grosse Schwierigkeit besteht, die verschieden Algorithmen auf einen gemeinsamen Nenner zu bringen. Dies wäre jedoch notwendig um eine einfache GUI Implementierung zu realisieren. Daher werden wir ein komplexeres vordefiniertes Labyrinth erstellen und dieses mit dem Right-Hand und einem weiteren intelligenteren Algorithmus lösen lassen.

Erster Lösungsalgorithmus implementieren (13 Stunden geschätzt)

- Right-Hand Algorithmus implementieren (Ben)

Lösung im GUI anzeigen (5 Stunden geschätzt)

- Mit einer anderen Farbe als schwarz die Lösung ins bestehende Labyrinth zeichnen (Ben)

GUI Optimierungen (3 Stunden)

- Debug Mode für GUI, der auf der Konsole das Labyrinth als Array ausgibt (Dominic)

Verbessern der Algorithmen (5 Stunden geschätzt)

- Ein und Ausgänge zeichnen beim Depth First Search Alge (Dominic)
- ~~Prim Algorithmus vervollständigen (Dominic)~~

3. Iteration (28.04.2013 – 25.05.2013)

In der dritten Iteration fokussierten wir uns klar auf die Fertigstellung des Projektes. Um unser Ziel zu erreichen mussten wir noch einen weiteren Lösungsalgorithmus implementieren und diverse kleinere Anpassungen am GUI vornehmen.

Wir stellten fest, dass wir für das interaktive Zeichnen im GUI unser Programm ändern müssen. Nach langer und intensiver Recherche stand fest, dass wir das Observer Pattern implementieren müssen, um diese Funktionalität gewährleisten zu können. Dies Stellte sich jedoch nochmals als eine grosse Hürde heraus. Um das Observer Pattern korrekt anwenden zu können muss auch auf das MVC Pattern zurück gegriffen werden. Leider hatten wir dieses Pattern zu diesem Zeitpunkt noch nicht korrekt implementiert, und so musste ein weiteres Refactoring der Applikation gemacht werden.

Die Lösung interaktiv im GUI darstellen (13 Stunden geschätzt)

- Die Lösung Punkt für Punkt zeichnen (Ben)
- Geschwindigkeit für das Zeichnen ist einstellbar (Ben)
- Labyrinth zentriert zeichnen (Ben)

Weitere Algorithmen implementieren (13 Stunden geschätzt)

- A-Star Lösungsalgorithmus implementieren (Ben)

Dokumentation erstellen (8 Stunden geschätzt)

- Grundgerüst erstellen (Dominic)
- Depth-First-Search detailliert beschreiben (Dominic)
- Lösungs-Algorithmus Right-Hand beschreiben(Ben)
- Lösungs Algorithmus A-Star beschreiben (Ben)

Schlusswort und Fazit

Erweiterungsmöglichkeiten

Fazit

Schlusswort

Projekt Links

Github Project Site	https://github.com/do3meli/Labyrinth
Jenkins Build Server	http://travelbutlr.com:8080/job/Labyrinth/
Sonar Analyse	http://rob.nerdherd.ch:9000/dashboard/index/269
Online Javadoc	http://travelbutlr.com:8080/job/Labyrinth/javadoc/

Literaturverzeichnis

Wikipedia Labyrinth Allgemein Stand: April 2013
http://de.wikipedia.org/wiki/Labyrinth

Wikipedia Maze Solving Algorithm Stand: May 2013
http://en.wikipedia.org/wiki/Maze_solving_algorithm

Depth-First-Search, Maze Algorithm Stand: November 2011
http://www.migapro.com/depth-first-search/

Abbildungsverzeichnis