

FLOODPIPE Programmierpraktikum

Lienau, John - Informationstechnischer Assistent (104923)



3. Dezember 2022

PTL-WEdel

Feldstraße 143, 22880 Wedel

Inhaltsverzeichnis

[2 Anforderungen 2](#_Toc120969681)

[2.1 Belegung des Spielfeldes 2](#_Toc120969682)

[2.2 Bedienung des Spiels 2](#_Toc120969683)

[2.3 Editor 2](#_Toc120969684)

[2.4 Speichern und Laden 2](#_Toc120969685)

[2.5 Zielgruppe 3](#_Toc120969686)

[3 Benutzerhandbuch 4](#_Toc120969687)

[3.1 Themen 4](#_Toc120969688)

[3.1.1 Thema 1 4](#_Toc120969689)

[3.2 Ablaufbedingungen 4](#_Toc120969690)

[3.3 Programminstallation 4](#_Toc120969691)

[3.4 Programmstart 5](#_Toc120969692)

[3.5 Bedienungsanleitung 5](#_Toc120969693)

[3.6 Fehlermeldungen 5](#_Toc120969694)

[3.7 Wiederanlaufbedingungen 5](#_Toc120969695)

[4 Programmierhandbuch 5](#_Toc120969696)

[4.1 Entwicklungskonfiguration 5](#_Toc120969697)

[4.2 Problemanalyse und Realisation 6](#_Toc120969698)

[4.3 Beschreibung grundlegender Datenstrukturen 7](#_Toc120969699)

[4.4 Programmorganisationsplan 7](#_Toc120969700)

[4.5 Programmtests 7](#_Toc120969701)

# Anforderungen

In Floodpipe soll implementiert werden, dass der Spieler ein Spielfeld mit verschiedenen Rohren sieht. Durch das Klicken auf eine Zelle kann der Spieler das Rohr drehen, um Verbindungen herzustellen. Das Ziel des Spiels ist es, alle Rohre so zu verbinden, dass keine offenen Enden vorhanden sind und jedes Rohr von der Quelle aus geflutet wird.

Programmiert wird mit der Delphi-IDE in der Programmiersprache Pascal, mit Unterstützung der Delphi SDK.

Im Folgenden sind die einzelnen gewünschten Elemente des Spiels:

## Bedienung des Spiels

Wenn der Spieler auf ein Rohrstück mit der linken Maustaste klickt, dreht sich das Rohrstück nach links um 90 Grad. Ein Rechtsklick dreht das Rohrstück im Uhrzeigersinn um 90 Grad. Nach jeder Drehung werden die Rohrstücke sichtbar, die von der Quelle aus gefüllt werden und diejenigen, die abgeschnitten sind. Die Füllung der Rohre erfolgt animiert. Von der Quelle ausgehend wird jeweils ein füllbares Rohrstück die Farbe ändern, bis keine weiteren Rohrstücke mehr gefüllt werden können. Die Geschwindigkeit der Animation kann vom Benutzer im Optionsmenü angepasst werden. Wenn alle Rohrstücke an ihren offenen Seiten verbunden sind, wird dem Spieler angezeigt, wie viele Klicks er zum Lösen des Rätsels benötigt hat. Anschließend startet ein neues Spiel.

## Belegung des Spielfeldes

Das Spielfeld besteht aus Zellen, die entweder mit Rohrstücken oder Mauerstücken belegt sind. Die Rohrstücke umfassen Geraden, Kurven, T-Verzweigungen und Endstücke, während maximal 10% der Zellen mit Mauerstücken belegt, sein dürfen. Das Spielfeld wird bei jedem Spielstart von einem Programm erstellt und gemischt, so dass die Rohrstücke in zufälliger Anordnung erscheinen. Die Quelle des Spiels befindet sich an einer zufälligen Position auf dem Spielfeld

## Editor

Im Editormodus werden neben dem Spielfeld die Rohrstücke, ein Mauerstück und eine Quelle angezeigt, die vom Nutzer auf dem Feld platziert werden können. Eine bereits vorhandene Zellenbelegung kann überschrieben werden, indem der Nutzer ein anderes Element auf dieselbe Zelle setzt. Die Quelle kann nur auf ein durch ein Rohrstück belegtes Feld gesetzt werden und darf nur einmal auf dem Spielfeld vorhanden sein. Sobald eine Quelle platziert ist, füllen sich die Rohre wie im Spielmodus, sodass der Nutzer das gelöste Spielfeld erkennen kann. Das Feld kann jederzeit gemischt werden, indem jede Zelle um eine zufällige Anzahl von Drehungen rotiert wird.

## Speichern und Laden

Das Programm bietet außerdem die Möglichkeit, das aktuelle Spiel in einer Datei zu speichern und zu einem späteren Zeitpunkt wieder zu laden. Hierbei wird ein Dateiname angegeben und wohin gespeichert oder geladen werden soll.

## Zielgruppe

Dieses Spiel könnte für Nutzer aller Altersgruppen geeignet sein, die gerne Rätsel lösen und ihr logisches Denken verbessern möchten. Es könnte auch für Nutzer interessant sein, die an der Erstellung und dem Design von Spielfeldern interessiert sind, da der Editormodus eine Möglichkeit bietet, eigene Spielfelder zu erstellen und zu bearbeiten.

Insgesamt bietet Floodpipe spannende Herausforderungen und kreative Möglichkeiten für Spieler jeden Alters.

# Benutzerhandbuch

Das Benutzerhandbuch besteht mindestens (ohne Ausnahme) ausfolgenden Kapiteln (Kapitelüberschriften müssen **genau so** beibehalten werden):

## Themen

We have now added a title, author and date to our first LaTeX document!

Hello World! My first Latex article

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi necante...

### Thema 1

Praesent imperdietmi nec ante. Donec ullamcorper, felis non sodales...

## Ablaufbedingungen

Die Mindestanforderungen, um ein Spielbares Erlebnis zu bekommen, sind folgende:

|  |  |
| --- | --- |
| Betriebssystem | Windows 11, 22H2 |
| CPU | Ryzen 9 7950x |
| GPU | NVIDIA GeForce RTX 4090 |
| RAM | 128GB @ 7GHz |
| FREIER SPEICHERPLATZ | 64GB |

## Programminstallation

Das Programm ist Stand Alone, das heißt, dass lediglich die Floodpipe.exe an beliebigen Orten ausgeführt werden kann. Sollten mit dem Programm Spielstände mitgegeben werden, so muss bekannt sein, wo diese gespeichert sind, um diese im Spiel auswählen zu können.

## Programmstart

Um das Spiel zu starten, muss die Floodpipe.exe gestartet werden. Anschließend sollte das bekannte Feld mit einem bereits Spielbaren Feld belegt sein.

## Bedienungsanleitung

Das Spiel startet mit einem Default Random-Game, welches bei jedem Neustart des Programms zufällig generiert wird und bereits spielbar erscheint. Zusehen sind nun Spielfeld und die Seiten-Knöpfe, eingeteilt in Spieleinstellungen und Editor Flächen.

Im Folgenden sind die einzelnen Komponenten beschrieben:

Der "Exit Button" beendet das Spiel und führt den Benutzer zurück zum Hauptmenü oder zum Desktop, je nachdem, wie das Spiel eingestellt ist. Dies kann nützlich sein, wenn der Benutzer das Spiel beenden und etwas anderes tun möchte.

### Knöpfe

Am rechten Rand des Spiels befinden sich Knöpfe, welche dem Benutzer frei zur Verfügung stehen.

#### Spiel/Editor Button

Der "Spiel/Editor Wechsel Button" ermöglicht es, zwischen dem normalen Spielmodus und dem Spiel Editor Modus hin und her zu wechseln. Im Editor Modus können die Rohre und anderen Elemente auf dem Spielfeld verändert und neu arrangiert werden, um neue Level zu erstellen (Siehe Editormodus).

#### Randomise Field Button

Der "Randomise Field Button" erlaubt es dem Benutzer, das aktuelle Spielfeld zufällig zu verändern. Dies kann nützlich sein, wenn der Benutzer eine neue Herausforderung sucht oder einfach nur in einem anderen Level spielen möchte.

#### Lade und Speicher Buttons

Die "lade und Speicher knöpfe" ermöglichen es dem Benutzer, den aktuellen Spielstand zu speichern und später wieder zu laden. Dies ist nützlich, wenn der Benutzer unterbrochen wird und das Spiel später fortsetzen möchte, oder wenn er einen bestimmten Spielstand speichern und später wiederholen möchte.

### Spielfeld

### Editor und Spiele Modus Wechsel

### Einstellungen

### Laden und Speichern von Spielständen

## Fehlermeldungen

Sämtliche Fehlermeldungen, die im Programm auftreten können, sind hier, bestehend aus den Spalten ’Fehlermeldung’, ’Fehlerursache’ und ’Behebungsmaßnahme’, aufgelistet.

|  |  |  |
| --- | --- | --- |
| Fehlermeldung | Fehlerursache | Behebungsmaßnahme |
| „You cant place a watersource there!” | Der Versuch eine Wasserquelle an einer Celle zu setzen, welche kein Rohr ist oder schon befüllt ist. | Nicht weiter versuchen eine Quelle an dieser Stelle zu setzen. |
|  |  |  |

## Wiederanlaufbedingungen

Was ist zu tun, wenn das Programm zur Laufzeit (z.B. durch Stromausfall) abgebrochen wird? Sind ungespeicherte Änderungen verloren gegangen? Welche Dateien sind betroffen?

# Programmierhandbuch

Grundsätzlich soll dieser Teil dazu dienen, einen Software-Entwickler über Deine Gedankengänge bei der Planung und Umsetzung des Projektes zu informieren, damit er sich nach möglichst kurzer Einarbeitungszeit der Fortführung, Pflege etc. widmen kann. Aussagekräftige und gut strukturierte Darstellungen sind hier gefordert. Stell Dir selbst einmal die Frage, was Du alles brauchst, um Dich in Deinem Projekt nach einigen Jahren wieder zurechtzufinden!

Das Programmierhandbuch ist der wichtigste Teil der Dokumentation und besteht immer und ohne Ausnahme mindestens aus folgenden Kapiteln (Kapitelüberschriften sollten genauso beibehalten werden):

## Entwicklungskonfiguration

Eine saubere und vollständige tabellarische Auflistung der Softwarekomponenten mit Hilfe derer Ihr das Programm entwickelt habt. Relevant sind hier Betriebssystem und Compiler mit Angabe der jeweiligen Version. Wurde das Programm auf mehreren unterschiedlichen Rechnern entwickelt, sind die unterschiedlichen Konfigurationen anzugeben.

## Problemanalyse und Realisation

Folgendermaßen können die erwarteten Informationen übersichtlich strukturiert werden (grundsätzlich gelten diese Ausführungen auch für andere / ähnliche technische Dokumentationen, in denen dem Leser Hintergrundwissen zur Lösung einer Aufgabenstellung vermittelt werden soll):

Sinnvollerweise startet man mit einer Auflistung der zu erfüllenden Aufgaben, wobei auf einen ausreichenden Detaillierungsgrad geachtet werden sollte. Die einzelnen Punkte ergeben sich aus der Aufgabenstellung (und werden ggf. in Diskussion mit dem Auftraggeber weiter verfeinert).

Die Aufstellung wird nach Sachgebieten, Modulen, übergeordneten Teilaufgaben o.ä. gruppiert und fasst ggf. Detailpunkte zusammen. Hiermit erhält man eine Art „Checkliste der zu erfüllenden Aufgaben“, die während der Projektphase durchaus noch modifiziert werden kann.

Damit verfügt man über ein Dokument (evtl. Teil eines Lasten- /Pflichtenheftes), das die wesentlichen Problemstellungen darstellt -> Problemanalyse.

In der nun folgenden Realisationsanalyse werden für jeden einzelnen Teilaspekt ein oder mehrere Lösungswege aufgezeigt und diskutiert (Pro und Contra). Das passiert häufig ohne Festlegung auf eine bestimmte Programmiersprache (hier im Programmierpraktikum gilt natürlich Delphi/Object Pascal!). In diesem Kapitel sollten also z.B. Fragen diskutiert werden wie ’ist eine iterative oder eine rekursive Lösung hier besser?’, ’ist Komponente x oder y aus Delphi besser geeignet?’ oder auch ’kann man diesen Punkt mit einem Array oder eine Liste besser umsetzen?’. Beschreibt hier immer objektive Vor- und Nachteile der verschiedenen Ansätze (und eben nicht ’das war für mich einfacher umzusetzen’).

Aus dieser Diskussion ergibt sich (hoffentlich) die beste aller Lösungen für diesen Problempunkt und man kann den nächsten anpacken… Sind nun alle einzelnen Punkte bearbeitet und steht damit ein Lösungsweg für jeden Teilaspekt fest, so ist das Kapitel „Realisationsanalyse“ (in der ersten Version) komplettiert.

Im dritten Unterkapitel steht für jede soeben gefundene (lt. Diskussion beste) Teillösung die Darstellung der Implementierung im Vordergrund. Charts, Diagramme, Verweise auf andere Kapitel (Programmorganisationsplan, Datenstrukturen etc.) helfen dem Softwareentwickler, der sich in Dein Projekt einarbeiten muss, Deine Gedankengänge nachzuvollziehen. Und schon ist die Realisationsbeschreibung fertig!

Die Präsentationsform des Gesamtkapitels obliegt natürlich Dir, sollte aber möglichst übersichtlich und wohlstrukturiert sein. Also beispielsweise: Zu lösendes Teilproblem – pro/contra zu den möglichen Lösungsansätzen – Beschreibung der Realisation. Und dann das gleiche noch einmal für die nächsten Punkte. Wählt man diese Darstellungsform, so entfällt für den Leser das leidige Springen zwischen den drei Hauptkapiteln. Zusätzlich kann dann auch die Überschriftenfolge „Problemanalyse“, „Realisationsanalyse“ und „Realisationsbeschreibung“ zu jedem Detailpunkt entfallen.

Unabhängig von der gewählten Präsentationsform sollte jeder Teilaspekt schnell zu finden sein: gute Struktur, keine übermäßig langen Texte, Konzentration auf die wesentlichen Fakten, ggf. Aufnahme in das Inhaltsverzeichnis, sinnvolle und eindeutige Überschriften. Eine erste Version dieses Kapitels ist tunlichst vor der eigentlichen Programmierung zu erstellen und als Planungshilfe für das Projekt zu benutzen. Natürlich können nach der Planung beim Programmieren Änderungen und Erweiterungen der ursprünglichen Problemanalyse und Realisation entstehen, die dann in die Endversion des Kapitels mit übernommen werden.

Um die Überlegungen auch nachvollziehbar zu machen ist eine gute Doku-Struktur mit passenden Überschriften unabdingbar (niemand liest 3-4 Seiten Fließtext).

## Beschreibung grundlegender Datenstrukturen

Hier werden alle Datenstrukturen, die Ihr im Programm tatsächlich benutzt, aufgelistet, jeweils mit einer abstrakten Beschreibung der Struktur (bei dynamischen Datenstrukturen wird zwecks besserer Anschaulichkeit zusätzlich eine Grafik verlangt), dem Zweck, dem diese Struktur in dem Programm dient, sowie Ihr pascal-spezifischer Aufbau (also die explizite Typdefinition aus dem Programm).

Außerdem wird der Aufbau aller im Programm verwendeten Dateien (sowohl typisierte Dateien als auch Textdateien) aufgeführt. Im Gegensatz zum Kapitel Problemanalyse und Realisation, in dem die Auswahl und deren Begründung einer oder mehrerer Datenstrukturen stattfindet, soll hier der tatsächliche Einsatz und die genaue Zusammensetzung der Datenstrukturen erläutert werden.

## Programmorganisationsplan

Stellt grafisch dar, wie sich die Units, die ihr verwendet, untereinander aufrufen. Es ist sinnvoll, in der Grafik die Units zu gruppieren (z.B. nach Formular, Logik, Daten, Typen, etc.). Dies kann zum Beispiel durch eine farbliche Hervorhebung geschehen. Zusätzlich zur Grafik sollte ein Fließtext oder eine tabellarische Darstellung die Grafik und ihre Aufteilung beschreiben.

## Programmtests

Während und nach der Programmerstellung muss das Programm umfangreich getestet werden! Dieser Vorgang soll nicht willkürlich geschehen. Dieser Testvorgang soll systematisch durchgeführt und dokumentiert werden.

Es müssen Testfälle für alle Funktionalitäten und alle Eventualitäten erdacht werden. Die Tests verlaufen also systematisch nach einem Vorgehensplan (für jedes Eingabefeld z.B. werden also alle möglichen Konstellationen inkl. Falscheingaben getestet).

Da das Programm von Euch selbst erstellt wurde, ist es Euch möglich, Testfälle abzuleiten, bei denen besondere Probleme erwartet werden. Es soll genügen, wenn nur diese speziellen Fälle dokumentiert werden (Programm kann blockieren, mögliche Dateninkonsistenzen, Verletzung von weiteren Bedingungen durch fehlerhafte Eingabe - z.B. ’Div by zero’). Dazu ist es aber notwendig, diese Fälle zu erkennen (am besten gleich beim Programmieren). Diese Testfälle stellt Ihr in diesem Kapitel in einer Tabelle, bestehend aus den Spalten ’Testfall’, ’erwartetes Ergebnis’ und ’erzieltes Ergebnis’ dar. In der Spalte ’Testfall’ steht jeweils eine detaillierte Beschreibung des Testfalls (so gut beschreiben, dass man ihn leicht nachstellen kann). In der Spalte ’erwartetes Ergebnis’ steht jeweils die Reaktion des Programms auf den Testfall, die Ihr erwartet habt (bei mathematischen Berechnungen also das Ergebnis), bevor Ihr den Testfall getestet habt. In der Spalte ’erzieltes Ergebnis’ steht das tatsächliche Ergebnis. Der Inhalt der beiden letzten Spalten wird, wenn alles gut geht, der gleiche sein. (Bei mathematischen Rechnungen können sich aufgrund der begrenzten numerischen Genauigkeiten Abweichungen ergeben, die nicht auf einen inkorrekten Algorithmus oder fehlerhafte Algorithmusumsetzung zurückzuführen sind. Diese sind dann ggf. kurz zu begründen.)

Werden zum Testen externe Dateien (schreibgschützte Datensätze, absichtlich veränderte Dateien, spezielle Spielstände etc.) benötigt, so sind diese in der Beschreibung des Testfalles klar zu benennen (und logischerweise mitzuliefern). Ist das Programm durch das Laden von Dateien in der Lage, bestimmte Zustände einzunehmen (z.B. Laden von besonderen Spielständen), so sollte mit verschiedenen Dateien die Funktionalität wichtiger Algorithmen demonstriert werden. Dieses Kapitel hat also zwei Aufgaben: Erstens soll es protokollieren, wie der Test organisiert und durchgeführt wurde, zweitens sollen die Testergebnisse dargestellt werden.