

FLOODPIPE Programmierpraktikum

Lienau, John - Informationstechnischer Assistent (104923)



11. Dezember 2022

PTL-WEdel

Feldstraße 143, 22880 Wedel

Inhaltsverzeichnis

[2 Anforderungen 2](#_Toc121667800)

[2.1 Bedienung des Spiels 2](#_Toc121667801)

[2.2 Belegung des Spielfeldes 2](#_Toc121667802)

[2.3 Editor 2](#_Toc121667803)

[2.4 Speichern und Laden 2](#_Toc121667804)

[2.5 Zielgruppe 3](#_Toc121667805)

[3 Benutzerhandbuch 4](#_Toc121667806)

[3.1 Ablaufbedingungen 4](#_Toc121667807)

[3.2 Programminstallation 4](#_Toc121667808)

[3.3 Programmstart 4](#_Toc121667809)

[3.4 Bedienungsanleitung 4](#_Toc121667810)

[3.4.1 Knöpfe 5](#_Toc121667811)

[3.4.2 Spielfeld 5](#_Toc121667812)

[3.4.3 Editor und Spiele Modus Wechsel 6](#_Toc121667813)

[3.4.4 Einstellungen 7](#_Toc121667814)

[3.4.5 Laden und Speichern von Spielständen 7](#_Toc121667815)

[3.5 Fehlermeldungen 9](#_Toc121667816)

[3.6 Wiederanlaufbedingungen 9](#_Toc121667817)

[4 Programmierhandbuch 10](#_Toc121667818)

[4.1 Entwicklungskonfiguration 10](#_Toc121667819)

[4.1.1 Software 10](#_Toc121667820)

[4.1.2 Hardware 10](#_Toc121667821)

[4.2 Problemanalyse und Realisation 11](#_Toc121667822)

[4.3 Beschreibung grundlegender Datenstrukturen 12](#_Toc121667823)

[4.3.1 Spieldatei 13](#_Toc121667824)

[4.4 Programmorganisationsplan 14](#_Toc121667825)

[5 Fazit 15](#_Toc121667826)

[6 Abbildungsverzeichnis 16](#_Toc121667827)

# 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

In diesem Handbuch werden Themen wie die Mindestanforderungen, Programminstallationen sowie alle Elemente des Spiels beschrieben.

## 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 |
| MAINBOARD | ASUSTeK COMPUTER INC. PRIME B650-PLUS |

## 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.

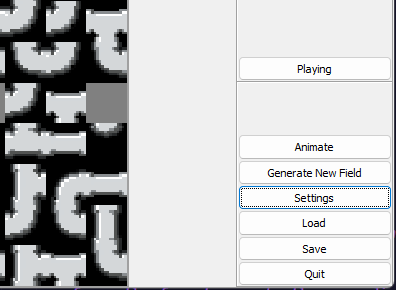


Abbildung 3‑1 Knöpfe des Spiels

#### 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

Das Spielfeld besteht aus 2 verschiedenen Kästchen. Den Wänden (in Grau) und den Rohren (welche verschiedene Variationen annehmen kann). Ein Spielfeld kann eine Größe zwischen 2x2 und 15x15 Kästchen annehmen, welche im Settings-Menu eingestellt werden.

Ein Bild, das Text, Licht enthält.

Automatisch generierte Beschreibung

Abbildung 3‑2 Das Spielfeld

### Editor und Spiele Modus Wechsel

Das Spiel kann zwischen zwei Modi wechseln: Editor- und Spielemodus.

Wenn sich das Spiel im Spielemodus befindet, so werden Klicks auf die Kästchen diese mit- oder gegen den Uhrzeiger drehen.

Im Editormodus wird der Button „Generate Field“ das Feld nichtmehr Mischen (Rohre drehen), und es erscheinen neue Knöpfe, welche eine Item-Auswahl darstellen. Klick der Nutzer auf diese, so kann er mit dem ausgewählten Item diese nun mit Klick auf ein Kästchen setzen.

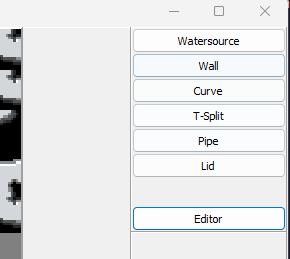


Abbildung 3‑3 Editor Modus

### Einstellungen

Wenn der Settings-Knopf gedrückt wird, erscheint ein Dialog, welches sowohl Spielfeld als auch die Animation einstellen kann.

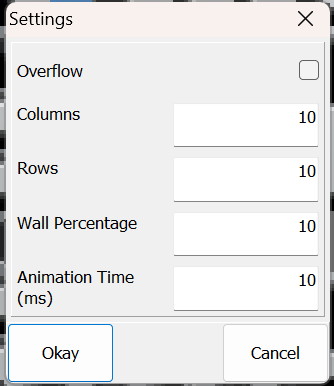


Abbildung 3‑4 Das Settings Menu

Zusehen ist der Overflow Modus und Animation Time, welche die Animation beeinflussen. Columns, Rows und Wall Percentage verändern das Spielfeld – beim klick auf Okay, wird der Nutzer beim Verändern dieser Werte gefragt, ob das Feld neu erstellt werden soll.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 3‑5 Änderung von Spielverändernden Einstellungen

Sollte dies nicht gewünscht sein, dann kann mit Klick auf „No“ dieser Vorgang abgebrochen werden. Sollte jedoch auf „Yes“ geklickt werden, so wird ein neues Spiel erstellt mit den gewählten Einstellungen. Es empfiehlt sich seinen vorherigen Spielstand zu Speichern.

Siehe Folgende Speicher und Lademöglichkeit:

### Laden und Speichern von Spielständen

Es gibt die Möglichkeit vorhandene Spielstände zu Speichern um diese zu einem späteren Zeitpunkt wieder zu Laden.

#### Speichern

Um ein Spiel zu Speichern muss der Knopf „Save“ gedrückt werden. Daraufhin wird ein Windows-Dialog erscheinen, welches Dateinamen und Speicherort bekommen möchte.

Standardmäßig wird das Fenster im Ort des Spiels sich öffnen und die Dateiendung wird .fpg (FloodpipeGame) sein.

Ein Bild, das Text, Screenshot, Monitor, Bildschirm enthält.

Automatisch generierte Beschreibung

Abbildung 3‑6 Windows Save Dialog

#### Laden

Wenn ein Spielstand geladen werden soll, so muss der „Load“ Knopf gedrückt werden. Sollte der Aktuelle Spielstand nicht Verloren gehen, wird hier vorher noch nach dem Speichern gefragt. Dann erscheint, wie beim Speichern, ein Windows Dialog um die Spieledatei auszuwählen.

Standartmässig wird nach der Endung .fpg gefiltert.

Ein Bild, das Text, Monitor, Screenshot, Bildschirm enthält.

Automatisch generierte Beschreibung

Abbildung 3‑7 Windows Load Dialog

## 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. |
| „No interaction durring simulation!“ | Versuchte Manipulation während einer Simulation | Nicht weiter versuchen während einer Simulation ein Spielfeld zu manipulieren. |
| „coulnd read from file“ | Eine Spieldatei lässt sich nicht laden. | Diese Datei einem Administrator zeigen und nach Fehlerursache suchen lassen. |
| „file doesnt exist“ | Es wird versucht eine Spieledatei zu laden, welche nicht existiert | Möglicherweise ist diese Spieledatei gelöscht worden, versuche es mit einer anderen. |
| “could not write to file” | Ein Spiel stand lässt sich nicht speichern | Es könnte versucht werden in schreibgeschützten Bereichen zu Speichern. Bitte versuche dies nicht unnötigerweise. Im besten Fall ist der Ort, an dem sich das Spiel befindet, ausreichend. |

## 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 ausfolgenden Kapiteln (Kapitelüberschriften sollten genauso beibehalten werden):

## Entwicklungskonfiguration

Entwickelt wurde an 2 Rechnern, welche die gleichen Softwarekonfigurationen, wie unten beschrieben, aufwiesen.

Folgend sind Soft- und Hardwarekomponenten aufgelistet:

### Software

|  |  |
| --- | --- |
| IDE | Embarcadero® Delphi 10.4 Version 27.0.40680.4203 |
| Compilerversion | 34.0 |
| Betriebssystem | Windows 11 Home Version 22H2 |
| System typ | 64-bit operating system, x64-based processor |

### Hardware

#### PC-1

|  |  |
| --- | --- |
| Prozessor | AMD Ryzen 9 7950X 16-Core Processor @ 5.8 GHz |
| Graphik | NVIDIA GeForce RTX 3070 @ 2 GHz |
| Ram | DDR 5 32,0 GB @ 6.4 GHz |
| Systemspeicher | PNY CS3140 1TB Gen. 4 M.2 SSD |
| Mainboard | ASUSTeK COMPUTER INC. PRIME B650-PLUS |

#### PC-2

|  |  |
| --- | --- |
| Prozessor | AMD Ryzen 5 1600 6-Core Processor @ 4.0 GHz |
| Graphik | NVIDIA GeForce RTX 1070 @ 1.68 GHz |
| Ram | DDR 4 16,0 GB @ 2.2 GHz |
| Systemspeicher | Crucial MX500 2.5 500 GB |
| Mainboard | MSI B350M PRO-VDH |

## 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

Alle im Programm verwendeten Datenstrukturen sind hier gelistet.

* TCellType ein Enum welches TYPE\_NONE, TYPE\_WALL und TYPE\_PIPE annehmen kann.
* TCellItem ein Enum welches PIPE\_LID, PIPE, PIPE \_TSPLIT und PIPE\_CURVES annehmen kann.
* TCellContent ein Enum mit CONTENT\_EMPTY und CONTENT\_WATER um den Inhalt anzugeben.
* TCEllRotation ein Enum mit NONE, FIRST, SECOND und THIRD um die Rotation anzugeben.
* TItemButton ein Enum um Button-Ids anzugeben.
* TPosition eine auflistung von zwei integer werten (x und y).
* PPositionNode ist ein Pointer auf eine TPositionNode.
* TPositionNode mit position (TPosition) und next (PPositionNode auf das nächste element).
* TPositionList besteht aus firstNode (PPositionNode) und lastNode (PPositionNode).
* PPipeTypeNode ist ein Pointer auf TPipeTypeNode.
* TPipeTypeNode besteht aus cellItem (TCellItem) cellRotation (TCellRotation) und next (PPipeTypeNode).
* TPipeTypeList besteht aus firstNode (PPipeTypeNode) und lastNode (PPipeTypeNode).
* TCell besteht aus einem image (TImage), cellType (TCellType), cellItem (TCellItem), cellContent (TCellContent), cellRotation (TCellRotation) und openings (TPositionList).
* TCellField besteht aus einem dynamischen Array von Array von TCell (x und y).
* TGameStruct besteht aus cellField (TCellField), cellRow- columnLength und wallPercentage (integer) und waterSourcePositionQueueList (TPositionList).

### Spieldatei

Die Spieledateien sind wie folgt strukturiert:

#### Header

Der Header ist bestehend aus Höhe, Breite und Wandprozent:

In der ersten Zeile steht die Höhe (Rows) des Spielfeldes.

In der zweiten Zeile steht die Breite (Columns) des Spielfeldes.

Dann in der dritten Zeile steht die Prozentzahl der Wände.

#### Dynamische Zeilen

Nun folgen die auf das Spielfeld angepassten Zeilen, welche aus den im Header beschriebenen Größen besteht.

Jede Zeile bekommt hierbei eine Zeile in der Datei. Jedes Feld ein aus den TCell bestehenden Typedefinitionen (TCellType, TCellItem, TCellContent und TCellRotation) als Zahl, Trennung erfolgt durch ein Leerzeichen.

Zum Schluss kommen die Wasserquellen bestehend aus x und y (X Y ) mit Leerzeichen getrennt.

Beispiel:

Ein Bild, das Elektronik enthält.

Automatisch generierte Beschreibung

Abbildung 4‑1 Spieledatei Beispiel

Zusehen ist eine Spieledatei mit einer Feldgröße von 2x10 einer Wandprozentzahl von 10% dem Feld und der Wasserquelle an Position 8x 0y.

## 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.

# Fazit

Ich habe während der Zeit des Praktikums sehr viele Interessante Berichte lesen können und habe ein sehr gutes Verständnis für wichtige Program-Planung entwickeln können.

Leider habe ich das zu spät bemerkt, wodurch ich nicht in der Lage war, Deadlines frühzeitig zu bemerken und für den Aufwand, den ich ins Programmieren gesteckt habe, gezielt zu nutzen. Was rauskam ist zwar nicht spielbar, dafür bin ich persönlich zufrieden, dass ich von 0 bis jetzt gelernt habe, mit verschiedensten Sprachen zu Programmieren.

Ich habe für die anstehende Abschlussarbeit direkt mit der Planung von Deadlines und der Analyse begonnen, sodass dies ein guter Ansatz werden kann. Wünschen tue ich mir, dass den Schülern an der PTL eine Art Großprojekt-Organisations-Plan (für Praktika und Abschlussarbeiten) beigebracht wird – welches Hilfreich sein kann, um sich an Deadlines zu halten, statt wie ich an Kleinigkeiten (was mich immer ausbremst). Ich habe dies erst im Praktikum bei Eyefactive lernen können, was zu spät für mich mit Floodpipe war, um da noch hinterher zu kommen mit der Konzentration.

Im Ganzen hat mir solch ein Projekt spaß gemacht und zeigt mir, dass ich noch viel Selbstdisziplin erlernen muss.

# Abbildungsverzeichnis

[Abbildung 3‑1 Knöpfe des Spiels 5](#_Toc121667828)

[Abbildung 3‑2 Das Spielfeld 6](#_Toc121667829)

[Abbildung 3‑3 Editor Modus 6](#_Toc121667830)

[Abbildung 3‑4 Das Settings Menu 7](#_Toc121667831)

[Abbildung 3‑5 Änderung von Spielverändernden Einstellungen 7](#_Toc121667832)

[Abbildung 3‑6 Windows Save Dialog 8](#_Toc121667833)

[Abbildung 3‑7 Windows Load Dialog 8](#_Toc121667834)

[Abbildung 4‑1 Spieledatei Beispiel 13](#_Toc121667835)