

Logo  
Projektpartner



## Diplomarbeit

# Prototyp: NeXt

Michael Leitner      Lukas Vogel

Imst, 25. Juni 2018

Betreut durch:  
Claudio Landerer

# **Eidesstattliche Erklärung**

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbst verfasst und keine anderen als die angeführten Behelfe verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht. Ich bin damit einverstanden, dass meine Arbeit öffentlich zugänglich gemacht wird.

---

Ort, Datum

---

Michael Leitner

---

Lukas Vogel

# **Abnahmeerklärung**

Hiermit bestätigt der Auftraggeber, dass das übergebene Produkt dieser Diplomarbeit den dokumentierten Vorgaben entspricht. Des Weiteren verzichtet der Auftraggeber auf unentgeltliche Wartung und Weiterentwicklung des Produktes durch die Projektmitglieder bzw. die Schule.

---

Ort, Datum

---

Auftraggeber

# **Vorwort**

Viele Informatiker spielen Computerspiele und so manch einer wünscht es sich auch eines selbst zu entwickeln. So ging es auch uns, glücklicherweise konnten wir diesen Wunsch durch diese Diplomarbeit verwirklichen. Wir entwickelten im Zuge dieses Projekts einen Prototyp für ein Spiel und lernten dabei einiges über die Spielentwicklung und welche Arbeiten gemacht werden müssen um ein Projekt dieser Größe zu bewerkstelligen.

Herzlich bedanken möchten wir uns bei unserem Betreuer Claudio Landerer und der Firma ClockStone, Innsbruck, die uns bei der Umsetzung unseres Projektes tatkräftig unterstützt haben.



# Abstract (Deutsch)

Thema:	Prototyp: NeXt
Name der Verfasser:	Michael Leitner & Lukas Vogel
Jahrgang:	2016/17
Schuljahr:	2017/18
Kooperationspartner:	ClockStone Softwareentwicklung GmbH
Aufgabenstellung:	Das Ziel dieser Diplomarbeit war es einen Prototyp für das Computerspiel NeXt, welches Elemente aus dem Puzzle sowie dem Jump and Run Genre beinhaltet, zu entwickeln. Besonders hervorzuheben ist das Einbauen des Time-Rift-Prinzips.
Realisierung:	Das Spiel wurde mit der Unity-Engine, welche auf der Programmiersprache C# basiert und mit der, vom selben Hersteller mitgelieferten, Entwicklungsumgebung verwirklicht. Für die Dokumentation verwendeten wir LaTeX in Kombination mit TeXstudio. Es kamen auch viele Techniken des Projektmanagements zur Anwendung.
Ergebnisse:	Als Resultat dieser Diplomarbeit entstand ein spielbares Computerspiel, welches die oben genannten Ziele erfüllt. Das Spiel hat 4 Level welche das Spielprinzip gut zur Geltung bringen. Die Dokumentation wurde erfolgreich abgeschlossen und beschreibt das gesamte Projekt.

Tabelle 0.1.: Abstract Deutsch



# Abstract (Englisch)

Topic:	Prototyp: NeXt
Authors:	Michael Leitner & Lukas Vogel
Year:	2016/17
Term:	2017/18
Cooperation Partners:	ClockStone Softwareentwicklung GmbH
Task:	The aim of this thesis was to create a prototype for the video game “NeXt”, which consists of elements of jigsaw games as well as the jump and run genre. Furthermore, the implementation of the “time-rift-principle” was of high importance to us.
Realization:	Das Spiel wurde mit der Unity-Engine, welche auf der Programmiersprache C# basiert und mit der, vom selben Hersteller mitgelieferten, Entwicklungsumgebung verwirklicht. Für die Dokumentation verwendeten wir LaTeX in Kombination mit TeXstudio. Es kamen auch viele Techniken des Projektmanagements zur Anwendung.
Conclusion/Result:	As a result of this thesis, a video game had been created, which matches all the operative goals we were aiming for. The game consists of 4 levels, showing off its principles to its advantage. Documentation has been completed successfully and it describes the project, the tasks given and additionally the realization.

Tabelle 0.2.: Abstract Englisch

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>12</b>
<b>Tabellenverzeichnis</b>	<b>13</b>
<b>Quelltexte</b>	<b>14</b>
<b>1. Einleitung</b>	<b>16</b>
<b>2. Projektmanagement</b>	<b>17</b>
2.1. Metainformationen . . . . .	17
2.1.1. Team . . . . .	17
2.1.2. Betreuer . . . . .	19
2.1.3. Partner . . . . .	20
2.1.4. Ansprechpartner . . . . .	20
2.2. Vorerhebungen . . . . .	21
2.2.1. Projektzieleplan . . . . .	21
2.2.2. Projektumfeld . . . . .	22
2.2.3. Risikoanalyse . . . . .	24
2.3. Pflichtenheft . . . . .	26
2.3.1. Zielbestimmung . . . . .	26
2.3.2. Produkteinsatz und Umgebung . . . . .	27
2.3.3. Funktionalitäten . . . . .	27
2.3.4. Testszenarien und Testfälle . . . . .	27
2.4. Planung . . . . .	28
2.4.1. Projektstrukturplan . . . . .	28
2.4.2. Meilensteine . . . . .	28
2.4.3. Gant-Chart . . . . .	28
2.4.4. Pläne zur Evaluierung . . . . .	28

2.4.5. Ergänzungen und zu klärende Punkte . . . . .	28
<b>3. Vorstellung des Produktes</b>	<b>29</b>
3.1. Produktbeschreibung . . . . .	29
<b>4. Eingesetzte Technologien</b>	<b>30</b>
4.1. Engine . . . . .	30
4.1.1. Spiel-Engine . . . . .	30
4.1.2. Unity-Engine . . . . .	30
4.2. Entwicklungsumgebung . . . . .	32
4.2.1. Unity-Editor . . . . .	32
4.2.2. Visual Studio . . . . .	32
4.3. C# . . . . .	32
4.4. Dokumentation . . . . .	32
<b>5. Problemanalyse</b>	<b>34</b>
5.1. USE-Case-Analyse . . . . .	34
5.2. Domain-Class-Modelling . . . . .	35
5.3. User-Interface-Design . . . . .	35
<b>6. Systementwurf</b>	<b>38</b>
6.1. Architektur . . . . .	38
6.1.1. Design der Komponenten . . . . .	38
6.1.2. Benutzerschnittstellen . . . . .	39
6.1.3. Datenhaltungskonzept . . . . .	39
6.1.4. Konzept für Ausnahmebehandlung . . . . .	39
6.1.5. Sicherheitskonzept . . . . .	39
6.1.6. Design der Testumgebung . . . . .	39
6.1.7. Desing der Ausführungsumgebung . . . . .	40
6.2. Detailentwurf . . . . .	40
<b>7. Implementierung</b>	<b>42</b>
<b>8. Tests</b>	<b>43</b>
8.1. Systemtests . . . . .	43
8.1.1. Einführung . . . . .	43
8.1.2. Testfälle . . . . .	43

8.2. Akzeptanztests . . . . .	43
<b>9. Projektevaluation</b>	<b>48</b>
9.1. Arbeitsaufwand . . . . .	48
9.2. Zusammenarbeit . . . . .	48
9.3. Zeitmanagement . . . . .	49
9.4. Produktevaluierung . . . . .	49
9.5. Kosten . . . . .	50
9.6. Erfahrungen . . . . .	50
9.7. Fazit . . . . .	50
<b>10. Zusammenfassung</b>	<b>51</b>
10.1. Resümee . . . . .	51
10.1.1. Resümee (Leitner Michael) . . . . .	51
10.1.2. Resümee (Vogel Lukas) . . . . .	52
<b>11. Beispielkapitel</b>	<b>54</b>
11.1. Beispiele zitieren . . . . .	54
11.1.1. Beispiele Abbildungen . . . . .	54
11.2. Beispiele Listen . . . . .	55
11.3. Beispiel Codesequenz . . . . .	57
11.3.1. Quicksort in JAVA . . . . .	57
11.4. Beispieltext . . . . .	59
<b>Literaturverzeichnis</b>	<b>61</b>
<b>A. Anhang-Kapitel</b>	<b>62</b>
A.1. Anhang-Section . . . . .	62

# **Abbildungsverzeichnis**

2.1.	Lukas Vogel . . . . .	17
2.2.	Michael Leitner . . . . .	18
2.3.	Mag. Claudio Landerer . . . . .	19
2.4.	ClockStone Softwareentwicklung GmbH . . . . .	20
2.5.	Michael Schiestl . . . . .	20
2.6.	Stakeholder-Diagramm . . . . .	23
2.7.	Matrix zur qualitativen Einordnung von Risiken . . . . .	24
2.8.	Projektstrukturplan . . . . .	28
5.1.	USE-Case-Diagramm . . . . .	34
5.2.	Wireframe-Hauptmenü . . . . .	35
5.3.	Wireframe-Levelmenü . . . . .	36
5.4.	Wireframe-Pausemenü . . . . .	37
11.1.	Arduino mit Lichtsensor und Lichterkette . . . . .	55

# **Tabellenverzeichnis**

0.1. Abstract Deutsch . . . . .	6
0.2. Abstract Englisch . . . . .	8
2.1. Stakeholder-Analyse . . . . .	22
2.2. Risiko-Bewertung-Maßnahmen . . . . .	26
8.1. Testfall 1: Bewegungssteuerung . . . . .	44
8.2. Testfall 2: Levelabschluss . . . . .	45
8.3. Testfall 3: Tod der Spielfigur durch Fall . . . . .	46
8.4. Testfall 4: Schalter zum Öffnen einer Tür . . . . .	47
11.1. Aufwändige Tabelle mit Abbildung und Caption . . . . .	56

# **Quelltexte**

11.1. QuickSort in Java . . . . . 57

# **Notationen**

Beschreibung wie Code, Hinweise, Zitate etc. formatiert werden

# **1. Einleitung**

Diese Diplomarbeit befasst sich mit der Entwicklung des Prototyps: NeXt. Das Produkt ist eine spielbare Version eines Computerspiels in dem sogenannte Jump and Run-Elemente, Puzzle-Elemente sowie Zeitmanagement-Elemente beinhaltet sind. Das Projekt wurde in Verbindung mit der Firma ClockStone Softwareentwicklung GmbH. erarbeitet. Interesse könnte diese Dokumentation bei jedem erwecken, der sich über Spielentwicklung in Verbindung mit der Unity-Engine informieren will. Des Weiteren werden auch die Aspekte des Projektmanagements dieser Arbeit dargelegt.

## 2. Projektmanagement

### 2.1. Metainformationen

#### 2.1.1. Team



Abbildung 2.1.: Lukas Vogel

**Name:** Lukas Vogel

**Funktion:** Projekt Leiter

**Wohnort:** Hohenems

**E-Mail:** lvogel@tsn.at

**Aufgabenbereiche:**

- GUI
- Leveledesign
- Dokumentation



Abbildung 2.2.: Michael Leitner

**Name:** Michael Leitner

**Funktion:** Projekt Leiter

**Wohnort:** Oberperfuss

**E-Mail:** mleitner@tsn.at

**Aufgabenbereiche:**

- Steuerung
- Tiem-Rift-Prinzip
- Dokumentation

### **2.1.2. Betreuer**

Seitens der Schule hat sich Herr Claudio Landerer bereiterklärt unser Projekt zu Betreuen. Er brachte uns in seinem Unterricht das Programmieren bei.



Abbildung 2.3.: Mag. Claudio Landerer

### **2.1.3. Partner**



Abbildung 2.4.: ClockStone Softwareentwicklung GmbH

Unser Partner während der Diplomarbeit war die Firma ClockStone Softwareentwicklung GmbH. ClockStone ist eine Spielentwicklungsunternehmen mit Sitz in Innsbruck, das im Jahr 2006 gegründet wurde. Einige ihrer Produkte wie zum Beispiel Bridge Constructor sind weltweit bekannt und sehr beliebt.

### **2.1.4. Ansprechpartner**

Unser Ansprechpartner des Unternehmens war Michael Schiestl. Er unterstützte uns bei Fragen bezüglich der Spielentwicklung und es war äußerst angenehm mit ihm zu arbeiten.



Abbildung 2.5.: Michael Schiestl

## **2.2. Vorerhebungen**

### **2.2.1. Projektzieleplan**

Projektziele-Hierarchie - SMART

## 2.2.2. Projektumfeld

### Stakeholder-Analyse

Die Ergebnisse der Stakeholder-Analyse zu sehen in Tabelle 2.1:

Stakeholder	Beschreibung	Einstellung	Nähe zum Projekt	Einfluss
Projektteam	Das Team ist dem Projekt positiv eingestellt und hofft, sich neues Wissen aneignen zu können.	sehr positiv	10	10
Landerer Claudio	Betreuer des Projekts.	neutral	9	10
ClockStone Softwareentwicklung GmbH.	Projektpartner der Diplomarbeit	positiv	9	10
Stefan Walch	Direktor der Schule und Studienkoordinator	neutral	5	10
Andere Lehrpersonen	Die anderen Lehrpersonen des IT-Kollegs sind während allen Präsentationen anwesend und können die Notengebung beeinflussen	neutral	2	4
Mitschüler	Manche Klassenmitglieder spielen gerne am Computer in der Freizeit	neutral bis positiv	1	1

Tabelle 2.1.: Stakeholder-Analyse

Die Einschätzung der Einstellung, der verschiedenen Stakeholder zum Projekt, in

Tabelle 2.1 geht von sehr negativ über neutral bis zu sehr positiv. Der Einfluss auf die Diplomarbeit wiederum wird mit einer Skala von 1-10 beschrieben, wobei 1 keinen bzw. sehr wenig Einfluss repräsentiert und 10 sehr starken. Auch die Nähe zum Projekt wird von 1-10 skaliert, 1 repräsentiert weit entfernt vom Projekt und 10 sehr nah.

### Stakeholder-Diagramm

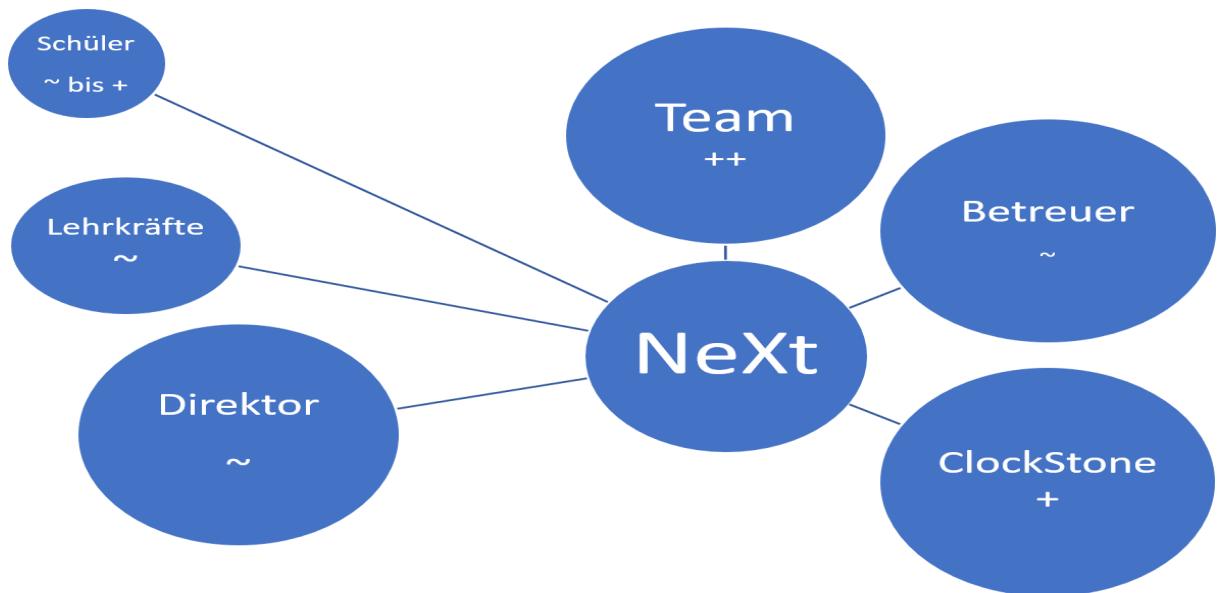


Abbildung 2.6.: Stakeholder-Diagramm

In diesem Diagramm, Abbildung 2.6 ist das Ergebnis der Tabelle 2.1 graphisch dargestellt. Die Größe der Kreise Zeigt den Einfluss, die Länge der Linien zwischen Stakeholder und NeXt die Nähe und die Zeichen (++ -> sehr positiv, -> neutral, --> sehr negativ) unter dem Namen des Stakeholders dessen Nähe zum Projekt.

### 2.2.3. Risikoanalyse

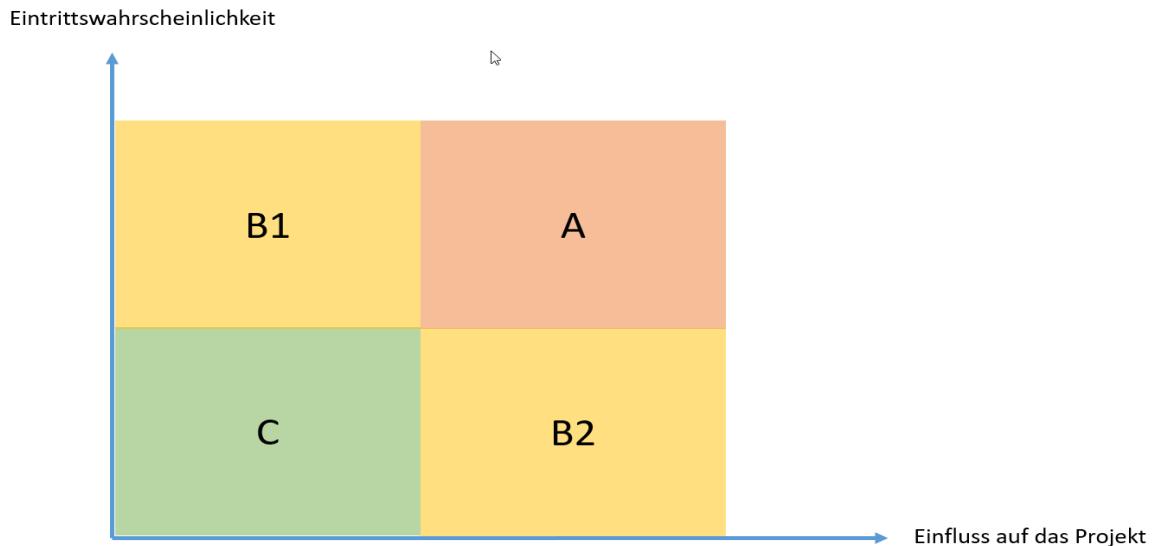


Abbildung 2.7.: Matrix zur qualitativen Einordnung von Risiken

Die Matrix (Abb. 2.7) wurde in vier Kategorien eingeteilt. Die Risiken werden dann zur jeweils passende Kategorie ein zugeordnet um diese dann besser einschätzen zu können. (Schwab, 2013, S. 141 -142)

Folgende Risiken sind aus der Projektumfeld-Analyse hervorgegangen und werden in Tabelle 2.2 beschrieben, bewertet und Maßnahmen gesetzt.

Risiko	Beschreibung	Bewertung	Maßnahme
Motivation des Teams	Die Motivation des Teams ist sehr wichtig. Wenn die sie sinkt, verringert sich auch die Arbeitsmoral, dies führt zu weniger Produktivität, was wiederum zum totalen Stillstand der Diplomarbeit führen kann.	A	Um die Wahrscheinlichkeit zu reduzieren sind am Anfang möglichst schnell Erfolgsergebnisse zu erzielen, dies steigert die Motivation sehr gut. Zusätzlich hilft es bei Problemen nicht Stunden lang am Problem zu sitzen, sondern entweder an etwas Anderem zu arbeiten oder jemanden um Hilfe zu bitten.
Zufriedenheit des Betreuers	Herr Landerer ist der Betreuer des Projekts und wenn er nicht mit unseren Leistungen zufrieden ist kann er das Projekt stoppen.	B2	<ul style="list-style-type: none"> <li>• laufende Berichterstattung</li> <li>• laufender Fortschritt im Projekt</li> <li>• bei Problemen oder Verzögerungen mit ihm reden</li> </ul>
Zufriedenheit des Partners	Die Firma ClockStone Softwareentwicklung GmbH soll wie der Betreuer mit dem Fortschritt und dem Projekt an sich zufrieden sein. Da auch sie im extrem Fall das Projekt zum Stopp zwingen könne.	B2	Auch hier kann eine laufende Berichterstattung und vor allem Meetings mit dem Partner helfen. Des weiteren ist zu beachten, dass auch ihr Unternehmen Spiele entwickelt, dadurch können sie bei Problemen kontaktiert werden, was zu einer schnelleren Lösungsfindung führen kann und somit auch ihren Wünschen entspricht.

Zufriedenheit des Direktors	Herr Walch ist auch für die Abnahme des Projekts zuständig. Wenn das Projekt seinen Anforderungen nicht entspricht kann er das Projekt abbrechen. Darüber hinaus ist er bei den Präsentationen der Diplomarbeit anwesend, dort werden von den allen Lehrern auch Fragen zum Projekt gestellt. Wenn diese nicht eine fachlich kompetente Antwort bekommen, kann das einen negativen Einfluss auf das Projekt haben.	B2	Um dieses Risiko zu abzuschwächen ist es besonders wichtig auf die Fragen und Anmerkungen von Herrn Walch einzugehen. Deswegen ist es essentiell sich gut auf die Präsentationen vor zu bereiten.
Zufriedenheit der anderen Lehrpersonen	Wie Herr Walch, sitzen auch andere Lehrpersonen während der Personen dabei und können Fragen zum Projekt stellen.	C	Dieses Risiko kann mittels guter Vorbereitung bei den Präsentationen verringert werden.

Tabelle 2.2.: Risiko-Bewertung-Maßnahmen

## 2.3. Pflichtenheft

### 2.3.1. Zielbestimmung

- Projektbeschreibung
- IST-Zustand
- SOLL-Zustand

- NICHT-Ziele (Abgrenzungskriterien)

### **2.3.2. Produkteinsatz und Umgebung**

- Anwendungsgebiet
- Zielgruppen
- Betriebsbedingungen
- Hard-/Softwareumgebung

### **2.3.3. Funktionalitäten**

- MUSS-Anforderungen
  - Funktional
  - Nicht-funktional
- KANN-Anforderungen
  - Funktional
  - Nicht-funktional

### **2.3.4. Testszenarien und Testfälle**

- Beschreibung der Testmethodik
- Testfall 1
- Testfall 2
- ...

## 2.4. Planung

### 2.4.1. Projektstrukturplan

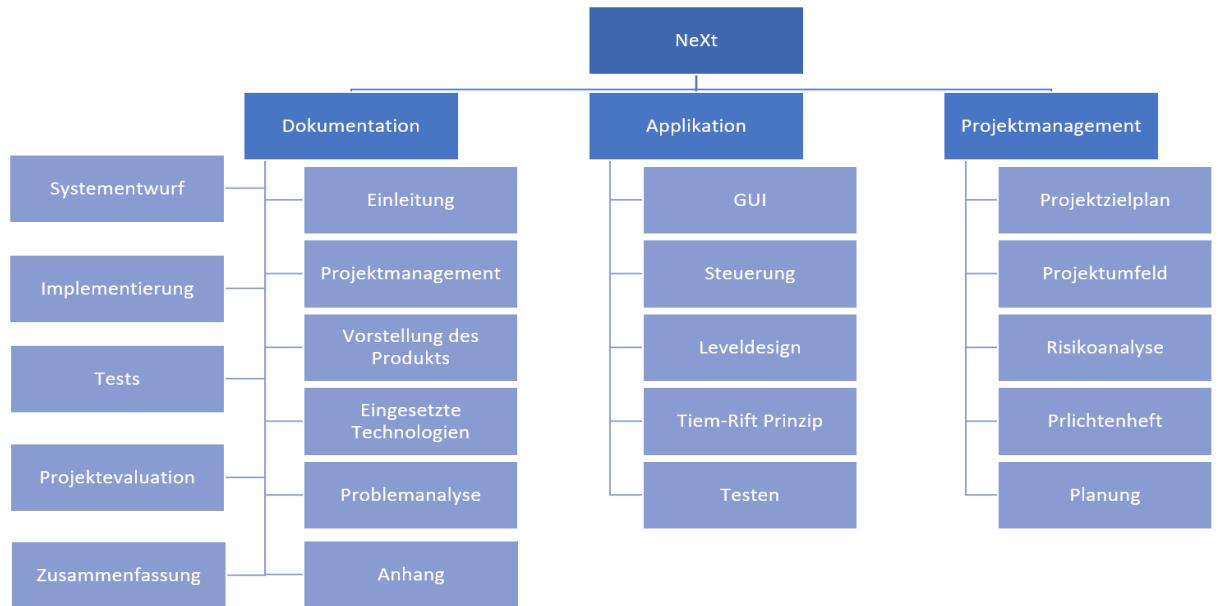


Abbildung 2.8.: Projektstrukturplan

Der Projektstrukturplan, zu sehen in Abbildung 2.8, soll den gesamten Umfang des Projekts strukturiert und gut lesbar darstellen.

### 2.4.2. Meilensteine

### 2.4.3. Gant-Chart

### 2.4.4. Pläne zur Evaluierung

### 2.4.5. Ergänzungen und zu klärende Punkte

# **3. Vorstellung des Produktes**

## **3.1. Produktbeschreibung**

Das Computerspiel NeXt ist in den Genres Jump and Run und Puzzle angesiedelt. Hauptaugenmerk dieses Spiels ist das sogenannte „Time-Rift“ Prinzip. Bei diesem Prinzip geht es darum, dass der Spieler jedes Level mehrmals in einzelnen Durchläufen spielen wird. Der Clue an der Sache ist jedoch, dass die zuvor gespielten Charaktere sich genauso bewegen wie sie in den vorigen Durchläufen gesteuert wurden. Das heißt: Hat der Spieler mit dem ersten Charakter einen Schalter betätigt, der eine Tür öffnet, dann wiederholt der Charakter diesen Vorgang, nur diesmal von selbst. Währenddessen kann der Spieler mit dem zweiten Charakter zu der zuvor genannten Tür gehen, welche sich öffnet, da der erste Charakter wieder den Schalter betätigt und somit wird das Level beendet. Durch dieses besondere Spielprinzip können vollkommen neue Puzzle-Elemente in das Produkt eingebaut werden, dies soll den Reiz des Spiels ausmachen. Damit das Spiel aber nicht zu leicht wird, wird noch ein Zeitfaktor eingebaut, welcher den Spieler unter Druck setzten soll. Schafft der Spieler nicht den Schalter in der geforderten Zeit zu betätigen, so wird er mit dem zweiten Charakter nicht durch die Tür kommen und kann deshalb nicht das Level beenden. Durch ein abwechslungsreiches Level Design und das oben beschriebene Spielprinzip soll ein interessantes Spiel entwickelt werden das auch einen großen Wiederspielfaktor als Ziel hat. Die Benutzeroberfläche so wie das Spielgefühl sollen intuitiv sein und für alle User keine große Umstellung zu anderen Spielen sein. Die genauen Mechaniken sowie die Funktionsweise des Spiels wurden in der Dokumentation erläutert.

# **4. Eingesetzte Technologien**

## **4.1. Engine**

### **4.1.1. Spiel-Engine**

Eine Spiel-Engine ist eine Art Framework, welches spezielle für Computerspiele entwickelt wurde. Die Engine ist zuständig für den Spielverlauf aber auch für die visuelle Darstellung des Spielablaufs. Viele Spiel-Engines liefern eine Entwicklungsumgebung mit, diese liefert die Werkzeuge für die Entwicklung der jeweiligen Applikation mit. Wikipedia (2018b)

### **4.1.2. Unity-Engine**

Auf anraten unseres Projektpartners verwendeten wir die Unity-Enigne, da das Unternehmen selbst damit arbeitet. Die Engine liefert eine Entwicklungsumgebung mit. Wir empfanden die Arbeit mit der Unity als angenehm. Zusätzlich ist es eine sehr weit verbreitete Spiel-Engine zu der es auch äußerst hilfreiches Lehrmaterial gibt. Unity selbst bietet auch eine Vielzahl an Lernvideos an.

Das Unternehmen Unity Technologies hat seine Hauptsitz in San Francisco und wurde 2004 gegründet. Die Zielplattformen für Unity sind PCs, Spielkonsolen, mobile Geräte sowie Webbrowser. Wikipedia (2018c)

## **Technische Eigenschaften**

Nachstehend werden die wichtigsten Technischen Eigenschaften der Unity-Engine erklärt.

### **Grafik**

Unity ist auf dem neusten Stand der Technik und unterstützt die verschiedensten Techniken die derzeit in der Spielentwicklung benötigt werden. Außerdem können die eingebauten Beleuchtungseffekte selbst bearbeitet werden. Dies erweitert die Möglichkeiten der Engine nochmals signifikant.

### **Animation**

Objekte des Spiels können über Skripte und andere Vorgehensweisen wie zum Beispiel physikalische Kräfte bewegt werden. Dieser Mechanismus war für unser Projekt sehr wichtig da der Spieler die Charaktere selbst in Echtzeit steuert.

### **Programmierung**

Unity unterstützt drei verschiedene Programmiersprachen:

- C#
- UnityScript (vergleichbar mit JavaScript)
- Boo

Diese sind notwendig um Skripte zu erstellen, welche die von Unity eingebauten Mechanismen erweitern. Da wir im Unterricht bereits mit C# Erfahrungen sammeln konnten, haben wir diese verwendet.

### **Werkzeuge**

Die Engine ist mittels Werkzeugen erweiterbar, diese sind zu einem Teil kostenlos Verfügbar und als einfache Plug-ins zu integrieren. Bei unserem Projekt konnten wir dadurch ohne Kosten einfache Grafiken und nützliche Tools einsetzen.

Wikipedia (2018c)

## **4.2. Entwicklungsumgebung**

### **4.2.1. Unity-Editor**

Als Entwicklungsumgebung wählten wir Unity-Editor da sie vom selben Unternehmen entwickelt wurde und somit speziell für die Unity-Engine gemacht wurde. Sie wurde basierend auf gängigen 3D-Animationsprogrammen designet. Sie ist sehr einfach zu bedienen und ermöglicht das importieren von Werkzeugen mittels eines einfachen Mausklicks. In dieser Entwicklungsumgebung können alle für die Spielentwicklung wichtigen Arbeiten verrichtet werden. Für das Programmieren der Scripte verwendeten jedoch wir Visual Studio. Wikipedia (2018c)

### **4.2.2. Visual Studio**

Wir verwendeten für die Programmierung von Scripten Visual Studio Community 2017, welche für Studenten kostenlos ist und wir im Unterricht zum entwickeln von Applikationen mit C# verwendeten.

## **4.3. C#**

Im Unterricht hatten wir unseren ersten Kontakt mit der Programmiersprache C#. Bei diesem Projekt konnten wir unser erlerntes Wissen nutzen, da wir es für die Scripte in Unity verwendeten.

## **4.4. Dokumentation**

Für die Dokumentation der Diplomarbeit verwendeten wir auf Anraten von unserem Betreuer LaTeX in Kombination mit der Applikation TeXstudio. LaTeX vereinfacht

*Prototyp: NeXt*

die Benutzung von TeX einem Textsatzsystem mittels Makros. TeXstudio ist eine Entwicklungsumgebung die für LaTeX entwickelt wurde. Wikipedia (2018a)

# 5. Problemanalyse

## 5.1. USE-Case-Analyse

In diesem Projekt wurde eine USE-Case-Analyse vollzogen. Als Benutzer gibt es nur den Anwender der Applikation, dessen Ziel es sein wird das Computerspiel zu spielen oder es zu beenden. Die USE-Case-Analyse und dessen Ergebnis wird als Diagramm in Abbildung 5.1 dargestellt.

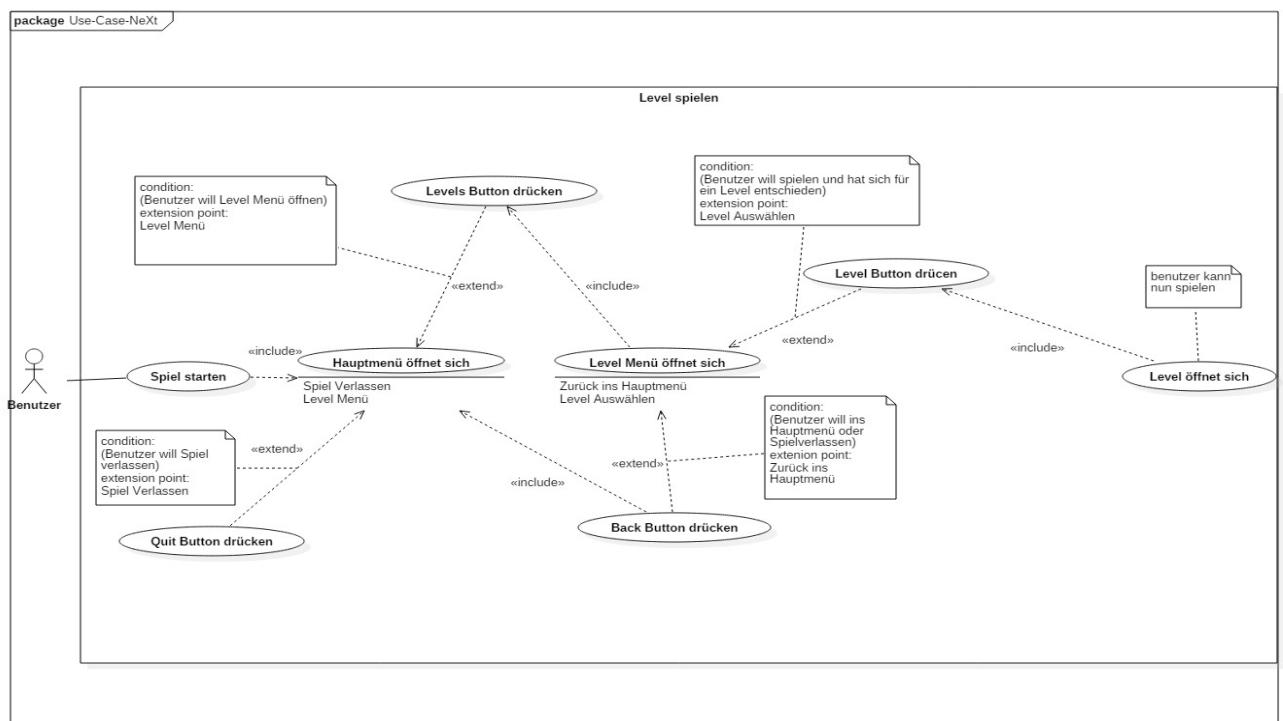


Abbildung 5.1.: USE-Case-Diagramm

## 5.2. Domain-Class-Modelling

- "Dinge" (Rollen, Einheiten, Geräte, Events etc.) identifizieren, um die es im Projekt geht
- ER-Modellierung oder Klassendiagramme
- Zustandsdiagramme (zur Darstellung des Lebenszyklus von Domain-Klassen darstellen)

## 5.3. User-Interface-Design

Im folgenden Teil der Dokumentation sind verschiedene Wireframes der Benutzeroberfläche des Programms zu sehen.

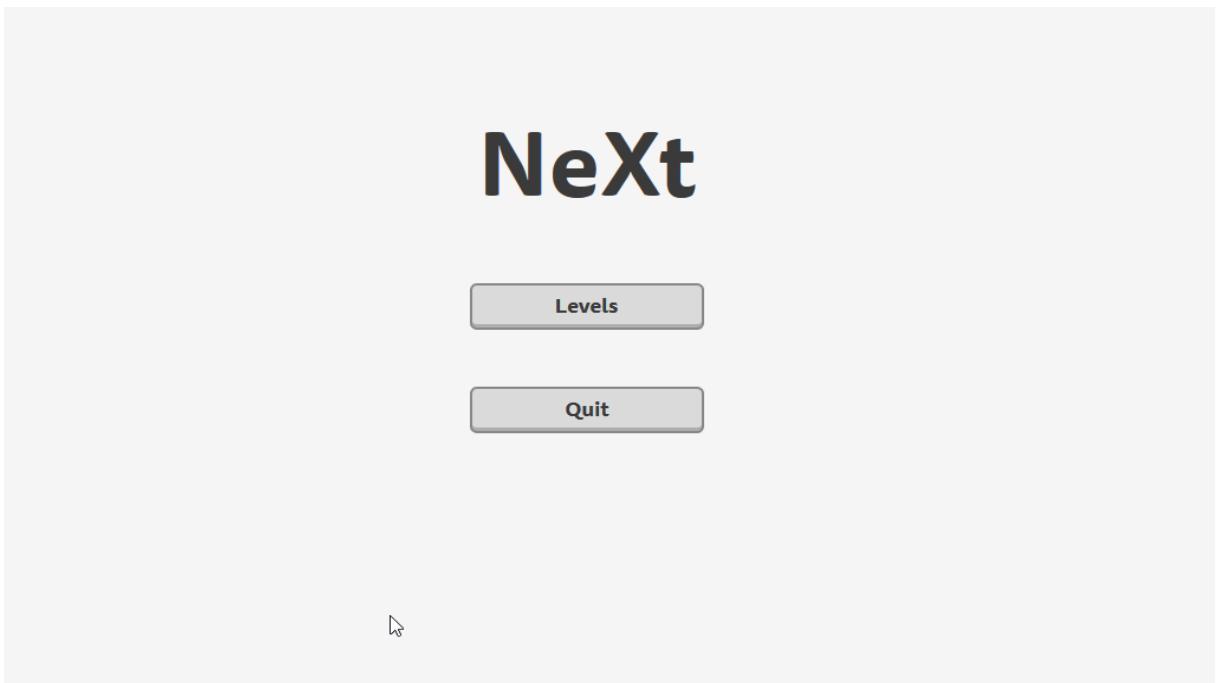


Abbildung 5.2.: Wireframe-Hauptmenü

In Abbildung 5.2 ist der erste Entwurf für das Hauptmenü des Spiels zu sehen. Diese Ansicht ist die Erste die der Benutzer sehen soll wenn er das Computerspiel startet.

## Prototyp: NeXt

Das Menü ist sehr einfach gehalten. Es gibt ein Textfeld welches "NeXt" anzeigt und darunter zwei Buttons. Der obere Button soll den Spieler in das Menü der Levels leiten, siehe Abbildung 5.3. Der untere Button soll die Applikation schließen.

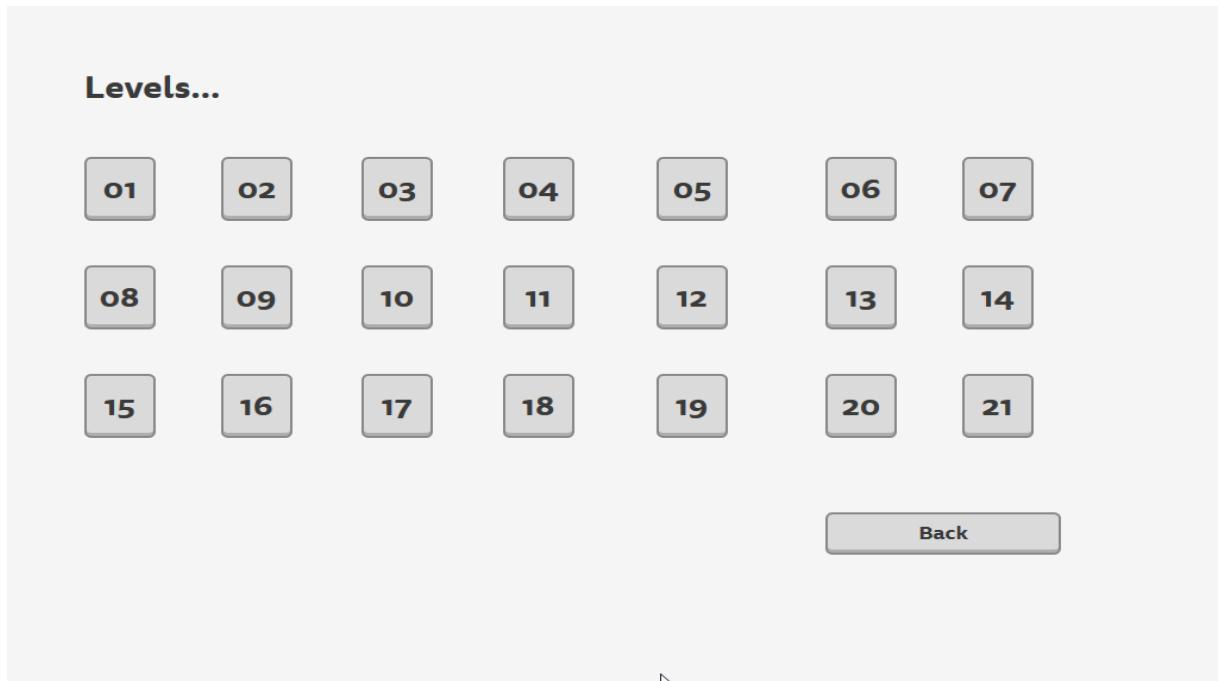


Abbildung 5.3.: Wireframe-Levelmenü

Das nächste Wireframe in Abbildung 5.3 zeigt eine Übersicht aller Spielbaren Levels welche als Nummerierte Buttons dargestellt werden. Wenn der Benutzer einen Level-Button betätigt soll sofort das gewählte Level starten. Der Button rechts unten mit der Beschriftung "Back" soll den Spieler wieder zurück in das Hauptmenü führen.

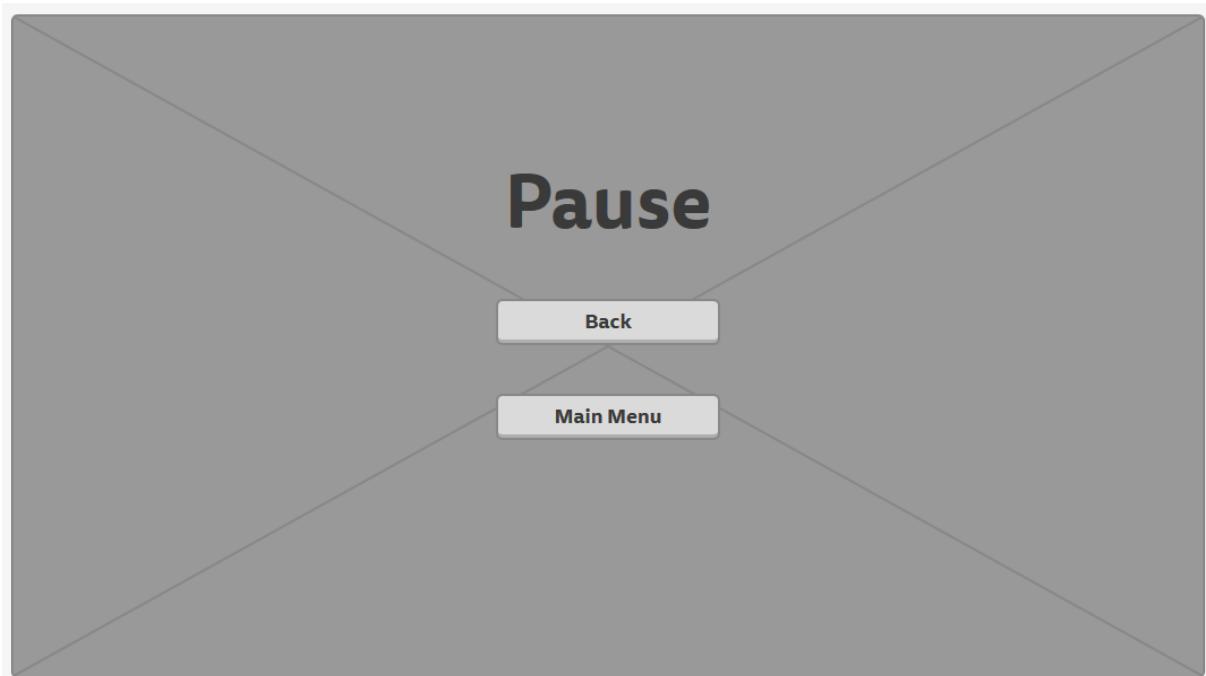


Abbildung 5.4.: Wireframe-Pausemenü

Sobald der Benutzer sich in einem Level befindet soll er die Möglichkeit haben, mittels Escape-Taste das Spiel stoppen und in das Pausemenü wechseln. Das Wireframe dazu ist in Abbildung 5.4 zu sehen. Die graue Fläche soll dabei das Level darstellen welches im Hintergrund immer noch zu sehen sein soll. Zusätzlich sind noch zwei Buttons zu sehen der erste Button mit dem Text "Back" soll das Menü wieder schließen und das Spiel weiter laufen lassen. Der Zweite welcher mit der Beschriftung "Main Menu" versehen ist soll dann Benutzer wieder in die Ansicht des Hauptmenüs, Abbildung 5.2 bringen.

# **6. Systementwurf**

## **6.1. Architektur**

### **6.1.1. Design der Komponenten**

Darstellung und Beschreibung der Systemarchitektur;

- statische Zerlegung des Systems in seine physischen Bestandteile (Komponenten, Komponentendiagramm)
- (textuelle) Beschreibung des dynamischen Zusammenwirkens aller Komponenten
- (textuelle) Beschreibung der Strategie für die Architektur, d. h. wie die Architektur in Statik und Dynamik funktionieren soll.
- Verwendung von Referenzarchitekturen bzw. Architekturmustern (als Schablonen, z.B. MVC, Plugin, Pipes and Filters)
  - MVC
  - Schichten
  - Pipes
  - Request Broker
  - Service-Oriented

### 6.1.2. Benutzerschnittstellen

- Design des UIs
- Dialoge, Dialogsteuerung, Ergonomie, Gestaltung, Eingabeüberprüfungen

### 6.1.3. Datenhaltungskonzept

- Design der Datenbank (ER-Modell)
- Design des Zugriffs auf diese Daten (Datenhaltungskonzept)
- Caching, Transaktionen

### 6.1.4. Konzept für Ausnahmebehandlung

- Systemweite Festlegung, wie mit Exceptions umgegangen wird
- Exceptions sind primär aus den Bereichen UI, Persistenz, Workflow-Management

### 6.1.5. Sicherheitskonzept

Beschreibung aller sicherheitsrelevanten Designentscheidungen

- Design der Security-Elemente
- Design von Safety-Elementen (Fehlertoleranz, Verfügbarkeit etc.)

### 6.1.6. Design der Testumgebung

- wie wird getestet (Unit-Testing, Integrationstesting, Systemtests, Akzeptanztests)

- Testumgebung, Testprozess, Teststrategie, Testmethoden, Testfälle

### **6.1.7. Desing der Ausführungsumgebung**

- Deployment (DevOps)
- Betrieb (besonders Hoch- und Hertunerfahren der Anwendung)

## **6.2. Detailentwurf**

Design jedes einzelnen USE-Cases

- Design-Klassendiagramme vom Domain-Klassendiagramm ableiten (incl. detaillierter Darstellung und Verwendung von Vererbungshierarchien, abstrakten Klassen, Interfaces)
- Sequenzdiagramme vom System-Sequenz-Diagramm ableiten
- Aktivitätsdiagramme
- Detaillierte Zustandsdiagramme für wichtige Klassen

Verwendung von CRC-Cards (Class, Responsibilities, Collaboration) für die Klassen

- um Verantwortlichkeiten und Zusammenarbeit zwischen Klassen zu definieren und
- um auf den Entwurf der Geschäftslogik zu fokussieren

Design-Klassen für jeden einzelnen USE-Case können z.B. sein:

- UI-Klassen
- Data-Access-Klassen
- Entity-Klassen (Domain-Klassen)
- Controller-Klassen

*Prototyp: NeXt*

- Business-Logik-Klassen
- View-Klassen

Optimierung des Entwurfs (Modularisierung, Erweiterbarkeit, Lesbarkeit):

- Kopplung optimieren
- Kohäsion optimieren
- SOLID
- Entwurfsmuster einsetzen

# **7. Implementierung**

## **7.1. GUI**

## **7.2. Leveledesign**

- GUI-Implementierung
- Controllerlogik
- Geschäftslogik
- Datenbankzugriffe

Detaillierte Beschreibung der Teststrategie (Testdriven Development):

- UNIT-Tests (Funktional)
- Integrationstests

Zu Codesequenzen:

- kurze Codesequenzen direkt im Text (mit Zeilnummern auf die man in der Beschreibung verweisen kann)
- lange Codesequenzen in den Anhang (mit Zeilennummer) und darauf verweisen (wie z.B. hier ??)

# **8. Tests**

## **8.1. Systemtests**

### **8.1.1. Einführung**

Nachfolgend sind 4 tabellarisch dargestellte Testfälle beschrieben, welche grundlegende Funktionen des Projekts Darstellen und deren Funktionalität gewährleisten sollen.

### **8.1.2. Testfälle**

## **8.2. Akzeptanztests**

<b>Nummer</b>	1
<b>Name</b>	Bewegungssteuerung
<b>Beschreibung</b>	Es soll gewährleistet werden, dass die Steuerung der Spielfigur mittels der unten angegebenen Tasten sowie deren Kombination miteinander wie gewünscht funktioniert.
<b>Vorbedingung</b>	<ul style="list-style-type: none"> <li>• Spiel muss gestartet sein</li> <li>• Level muss gewählt sein</li> </ul>
<b>Tester</b>	Lukas Vogel
<b>Datum</b>	8.5.2018
<b>Vorgehen</b>	<p><i>Soll-Verhalten:</i></p> <ul style="list-style-type: none"> <li>• A → Bewegung der Spielfigur nach links.</li> <li>• D → Bewegung der Spielfigur nach rechts.</li> <li>• Leertaste → Vertikale Bewegung der Spielfigur nach oben.</li> <li>• A + Leertaste → Eine gemischte Bewegung die Horizontal nach links und Vertikal nach oben geht (Sprung nach links).</li> <li>• D + Leertaste → Eine gemischte Bewegung die Horizontal nach rechts und vertikal nach oben geht (Sprung nach rechts)</li> </ul> <p><i>Ist Verhalten:</i></p> <ul style="list-style-type: none"> <li>• A → Spielfigur bewegt sich nach links.</li> <li>• D → Spielfigur bewegt sich nach rechts.</li> <li>• Leertaste → Spielfigru bewegt sich vertikal nach oben.</li> <li>• A + Leertaste → Sprung nach links.</li> <li>• D + Leertaste → Sprung nach rechts.</li> </ul>
<b>Erfolgskriterien</b>	<p><i>Gewünschtes Verhalten:</i></p> <ul style="list-style-type: none"> <li>• Flüssige Bewegung ohne Stopps</li> <li>• Bedingte Beeinflussung des Verhaltens durch die Physik-Engine</li> <li>• Tatsächliche Bewegung durch Tastendruck</li> </ul>

Tabelle 8.1.: Testfall 1: Bewegungssteuerung

<b>Nummer</b>	2
<b>Name</b>	Levelabschluss
<b>Beschreibung</b>	Es soll gewährleistet werden, dass der Spieler beim Abschließen des Levels in eine Übersicht (Level-Übersicht) gebracht wird. Durch das öffnen der Level-Übersicht kann der User entscheiden ob er ein anders Level öffnen will oder mittels Button ins Hauptmenü der Anwendung.
<b>Vorbedingung</b>	<ul style="list-style-type: none"> <li>• Spiel muss gestartet sein</li> <li>• Level muss gewählt sein</li> <li>• Spieler muss Ende des Levels erreichen</li> </ul>
<b>Tester</b>	Lukas Vogel
<b>Datum</b>	8.5.2018
<b>Vorgehen</b>	<p><i>Soll-Verhalten:</i></p> <ul style="list-style-type: none"> <li>• Beim Levelabschluss kommt ein Overlay, das 2 Möglichkeiten bieten (Nächstes Level, Hauptmenü)</li> <li>• Wird einer der Levelbuttons (nummerierte Quadrate) betätigt, startet das gewünschte Level.</li> <li>• Der Button „Back“ bringt den Spieler in das Hauptmenü des Spieles.</li> </ul> <p><i>Ist Verhalten:</i></p> <ul style="list-style-type: none"> <li>• Wechsel in die Levelübersicht nach Abschluss des Levels</li> <li>• Wird Levelbutton gedrückt startet richtiges Level</li> <li>• Back-Button öffnet Hauptmenü</li> </ul>
<b>Erfolgskriterien</b>	<p><i>Gewünschtes Verhalten:</i></p> <ul style="list-style-type: none"> <li>• Der Spieler soll durch das Overlay die Auswahlmöglichkeit haben, weiter zu spielen oder zurück ins Hauptmenü zu navigieren.</li> </ul>

Tabelle 8.2.: Testfall 2: Levelabschluss

<b>Nummer</b>	3
<b>Name</b>	Tod der Spielfigur durch Fall
<b>Beschreibung</b>	Bei einem Fehler des Spielers, indem er durch falsches Steuern des Charakters aus der Spielwelt fällt, wird der Charakter wieder zur Startposition zurückgesetzt.
<b>Vorbedingung</b>	<ul style="list-style-type: none"> <li>• Spiel muss gestartet sein</li> <li>• Level muss gewählt sein</li> <li>• Spieler fällt aus der Spielwelt</li> </ul>
<b>Tester</b>	Michael Leitner
<b>Datum</b>	8.5.2018
<b>Vorgehen</b>	<p><i>Soll-Verhalten:</i></p> <ul style="list-style-type: none"> <li>• Charakter wird ab einer bestimmten Grenze („Deathzone“) an den Startpunkt des Levels zurückgesetzt („Respawn“).</li> </ul> <p><i>Ist Verhalten:</i></p> <ul style="list-style-type: none"> <li>• Respawn der Spielfigur funktioniert.</li> </ul>
<b>Erfolgskriterien</b>	<p><i>Gewünschtes Verhalten:</i></p> <ul style="list-style-type: none"> <li>• Durch das zurücksetzen des Charakters soll ein reibungsloses weiterspielen möglich sein.</li> </ul>

Tabelle 8.3.: Testfall 3: Tod der Spielfigur durch Fall

<b>Nummer</b>	4
<b>Name</b>	Schalter zum Öffnen einer Tür
<b>Beschreibung</b>	Beim Betätigen des Schalters mittels der Spielfigur, soll eine Tür/Passage geöffnet werden, welche für den weiteren Spielverlauf wichtig ist.
<b>Vorbedingung</b>	<ul style="list-style-type: none"> <li>• Spiel muss gestartet sein</li> <li>• Level muss gewählt sein</li> <li>• Spieler ist bei einem Schalter</li> </ul>
<b>Tester</b>	Michael Leitner
<b>Datum</b>	8.5.2018
<b>Vorgehen</b>	<p><i>Soll-Verhalten:</i></p> <ul style="list-style-type: none"> <li>• Spieler bewegt Spielfigur auf den Schalter</li> <li>• Tür öffnet sich, sobald der Schalter betätigt wurde.</li> </ul> <p><i>Ist Verhalten:</i></p> <ul style="list-style-type: none"> <li>• Spieler benutzt Schalter und Tür geht auf.</li> </ul>
<b>Erfolgskriterien</b>	<p><i>Gewünschtes Verhalten:</i></p> <ul style="list-style-type: none"> <li>• Durch das öffnen der Tür sollen neue Gebiete des Levels erschlossen werden, um auch letztendlich das Level abzuschließen zu können</li> </ul>

Tabelle 8.4.: Testfall 4: Schalter zum Öffnen einer Tür

# **9. Projektevaluation**

## **9.1. Arbeitsaufwand**

Während der Bearbeitung der Diplomarbeit, lernten wir schnell, dass wir den Arbeitsaufwand des Projektes unterschätzt hatten, am besten wäre es, wenn man sich während den verschiedenen Ferien mindestens für mehrere Tage getroffen hätte und dann intensiv gearbeitet hätte. Schlussendlich ist das Projekt zwar trotzdem fertiggestellt worden und es erfüllt auch die gewünschten Anforderungen, jedoch war so viel mehr Arbeit zusätzlich zum normalen Schultag nötig.

## **9.2. Zusammenarbeit**

Die Zusammenarbeit des Teams war nie ein Problem, die einzige Komplikation die wir hatten war das Ausfallen eines Teammitglieds. Dies führte zu einer erneuten Aufteilung der Aufgaben und zu einer Umstrukturierung des Projektteams. Als Resultat ist der Arbeitsaufwand stark gestiegen und das Projekt war ein noch größere Herausforderung.

Wichtig ist es noch anzumerken, dass bei der Zuordnung der Themen keine Probleme auf traten, da jeder im Team seinen Wunschschwerpunkt ohne Diskussion bekam. Dies ist darauf zurück zu führen, dass wir diesbezüglich unterschiedlich sind und bei diesem Projekt die Themenauswahl passend zu unseren Persönlichkeiten war.

## 9.3. Zeitmanagement

Die größte Schwierigkeit war das Zeitmanagement, da das gesamte Team sich nicht nur auf die Bearbeitung der Diplomarbeit konzentrieren konnte, sondern zusätzlich noch die benötigten schulischen Leistungen erbringen musste. Dies führte des Öfteren zu Komplikationen und somit auch zu zeitlichen Verschiebungen von Meilensteinen. Durch Teamarbeit und Arbeitsaufteilung konnte jedoch auch diese Herausforderung gemeistert werden und das Projekt dennoch mit ausreichender Zeit fertig gestellt werden.

## 9.4. Produktelevaluierung

### Sollzustand

Es soll ein Spiele-Prototyp entwickelt, der die vorgegebenen Funktionen realisiert. Das Spiel wird vollständig spielbar sein und einige Levels enthalten, die das Spielprinzip voll zur Geltung bringen. Die Software wird in einer Beta-Phase mit Usern getestet, deren Feedback zur Verbesserung des Spiels verwendet wird. Das Spiel wird als voll funktionsfähiges Spiel an den Auftraggeber übergeben.

### Istzustand

Es wurde ein Prototyp entwickelt, der alle vorgegebenen Funktionen realisierte. Das Spiel ist vollständig spielbar und ist mit 4 Level ausgestattet. In diesen Level kommt das grundlegende Spielprinzip zur Geltung. Es wurde auf die Beta-Phase verzichtet, aus dem Grund, dass keine Zeit mehr für einen durchlauf einer Beta-Testphase sowie die Einarbeitung der Ergebnisse des Feedbacks, war. Wegen dem Verlassen eines Projektmitgliedes wurde bei dem Spiel gänzlich auf Soundeffekte und Hintergrundmusik verzichtet. Ebenso wurde die Grafik sehr primitiv gehalten, da das Erstellen von neuen Grafiken sehr aufwendig ist und wir dafür eine zusätzliche Einarbeitungsphase benötigt hätten.

## **9.5. Kosten**

Kosten sind für das Projekt insofern keine entstanden, da wir die benötigte Software für die Entwicklung komplett kostenlos bekommen haben. Die einzigen Kosten waren Zeit und Arbeit somit sind keine Ausgaben entstanden.

## **9.6. Erfahrungen**

Besonders hervorzuheben ist die Vielfalt an neuen Erfahrungen, die wir während des Projektes sammeln konnten. Wir lernten den gesamten Umfang eines Projekts kennen und welche Arbeitsschritte für ein solches Unterfangen notwendig sind, sei es nur die ständige Kommunikation mit den Teammitgliedern bis zur Fertigung einer umfangreichen Projektdokumentation.

## **9.7. Fazit**

Abschließend ist zu sagen, dass es ein sehr spannendes Projekt war, welches uns nicht nur ein Mal forderte. Wir lassen uns die Möglichkeit offen NeXt noch weiter zu entwickeln und vielleicht sogar eine Marktfähige Version zu produzieren.

# **10. Zusammenfassung**

## **10.1. Resümee**

### **10.1.1. Resümee (Leitner Michael)**

Die Entwicklung eines Spieles war ein größerer Aufwand als erwartet.

Gründe dafür sind:

1. Für die ausgewählte Engine war eine intensivere Einarbeitung in die Dokumentation notwendig, um auch die Engine ohne Probleme verwenden zu können.
2. Der Absprung eines Projektmitgliedes hatte uns mehr Arbeit beschert, aber durch Absprache mit dem Projektpartner und Projektbetreuer, wurde der Projektumfang angepasst.
3. Das komplette Diplomprojekt neben dem normalen Schultag durchzuführen und erarbeiten, war ebenso ein Faktor, der sich schwer auf das Zeitmanagement gelegt hat.

Letztendlich ist aber alles im Projekt, mehr oder weniger, gut verlaufen. Des Weiteren habe ich viel für die Spiele Entwicklung gelernt, dass ich für zukünftige Projekte (ob privat oder beruflich) anwenden kann. Die Arbeitsaufteilung war zwischen Lukas Vogel und mir sehr gerecht und beide hatten ungefähr den gleichen Aufwand. Für alle die ein Spiel entwickeln möchten gebe ich nur einen Tipp auf dem Weg: „Plant euch genug Zeit ein. Es hat einen Grund, warum so viele Veröffentlichungen von Spielen verschoben werden.“

### **10.1.2. Resümee (Vogel Lukas)**

Während der Diplomarbeit lernte ich den Ablauf eines Projekts haut nah kennen und konnte somit viele wertvolle Erfahrungen sammeln, welche mir sicherlich auch in Zukunft helfen werden.

Die ersten Neuerungen für mich waren die vielen verschiedenen Programme, die wir im Laufe unsers Projekts verwendeten. Eine Software, die für mich komplett neu war, war Latex mit der wir die Diplomarbeit dann geschrieben haben. Ein Textverarbeitungsprogramm in der man die Formatierung „programmiert“ war etwas Ungewohntes, jedoch ist das Programm perfekt auf wissenschaftliche Arbeiten wie Diplomarbeiten zugeschnitten und zeigte uns einige Vorteile gegenüber anderer Applikationen.

Mitten während des Projekts lernten wir wie wichtig es ist immer einen Plan-B zu haben, da eine Teammitglied die das IT-Kolleg frühzeitig beendete und wir somit ein Teammitglied weniger waren. Dies hat die Folge mit sich das wir die Aufteilung sowie den Umfang des Projekts noch einmal komplett überdenken mussten.

Besonders interessant war das Kennenlernen der Spieleentwicklung und dadurch einen Überblick zu bekommen wieviel Arbeit hinter so einer Software steht. Das erste große Thema war die Einarbeitung in eine komplett fremde Programmierumgebung. Natürlich hatten wir mit der Programmiersprache C# schon während unser Unterrichts zu tun, jedoch ist das Programmieren mit C# in Kombination mit der Spiele-Engine Unity nochmal etwas ganz anderes. Was mir dabei aber auffiel war, dass nach der Lernphase und der Eingewöhnungszeit in die neue Entwicklungsumgebung von Unity, auch hier wieder einige Parallelen mit den im Unterricht besprochen Themen zu ziehen waren. Man darf aber an dieser Stelle nicht glauben, dass das entwickeln mit Unity dem normalen Programmieren von Anwendungen stark ähnelt, da Unity auf eine ganz eigene Art funktioniert.

Die größte Aufgabe war eindeutig das Zeitmanagement. Das Problem war, zusätzlich zu den vielen Unterrichtseinheiten und die Zeit, die für das Lernen oder das erledigen anderer wichtiger schulischer Tätigkeiten war, noch Zeit für das Projekt zu finden. Diese Schule fordert einen mehr als wie alle anderen die ich davor besucht habe und

*Prototyp: NeXt*

zusätzlich kommt dann noch die Diplomarbeit hinzu. Es gab nicht nur einmal den Punkt, an dem ich nicht mehr wusste wie ich alles erledigen sollte. Abschließend ist nur zu sagen das es eine besondere Herausforderung war diese Diplomarbeit zu erstellen und ich sehr viel fürs Leben dazugelernt habe.

# **11. Beispielkapitel**

## **11.1. Beispiele zitieren**

Wikipedia (2018b)

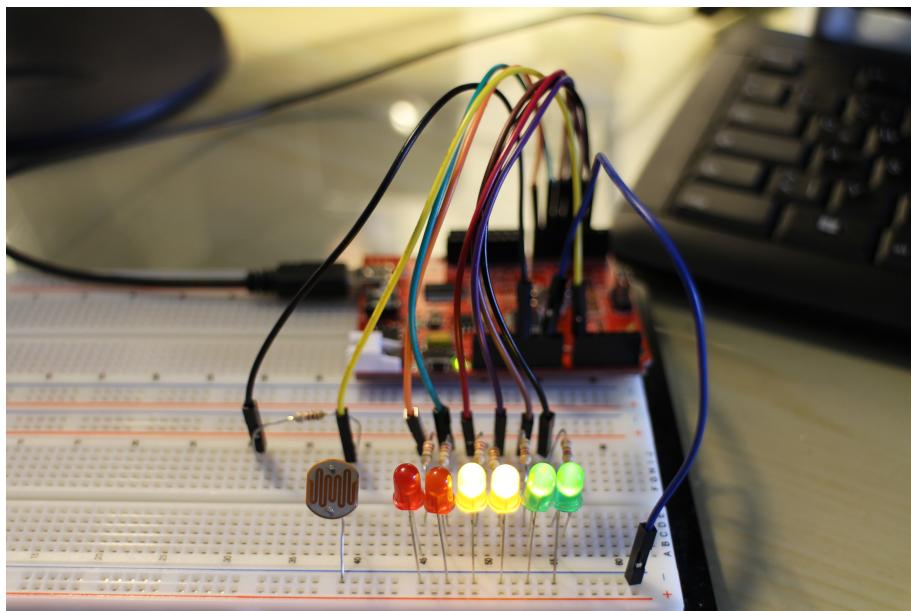
### **11.1.1. Beispiele Abbildungen**

Auf diese Weise kann man zum Beispiel in Latex auf die Abbildung 11.1 verweisen. Die Kennung für den Verweis vergibt man selbst mit dem „label“ Kommando bei der Abbildung.

Jede Abbildung muss nicht nur mindestens einen Verweis im Text haben. Es wird außerdem eine Bildunterschrift verlangt. Für diese ist festgesetzt, dass die Abbildungsunterschrift alleine ausreichend sein muss, um zu verstehen, was am Bild zu erkennen ist.

Der nächste wichtige Punkt sind die Quellenangaben bei Abbildungen. Der Author muss zu jeder Abbildung die notwendigen Rechte haben und idealer Weise gibt man diese bei der Abbildung mit an. In Abbildung 11.1 auf Seite 55 sieht man das.

Es ist wichtig zu verstehen, dass Latex die Positionierung von Abbildungen übernimmt. Man definiert die Abbildung über begin-figure dort, wo man die Abbildung in etwa haben möchte, den Rest übernimmt Latex



© Stefan Stolz (CC BY-SA 3.0)

Abbildung 11.1.: Hintergrund: Arduino Board; Vordergrund: eine Lichterreihe und ein Lichtsensor (Fotowiderstand); In diesem Beispiel wird die Lichterreihe je nach Helligkeit des Umgebungslichtes gesteuert. Durch leichte Modifikationen kann man damit eine Lichtschranke oder auch eine Helligkeitssteuerung für das Smartphone simulieren.

## Beispiele Tabellen

Tabelle 11.1 ist ein Beispiel für eine aufwändiger Tabelle mit einer Abbildung und Überschrift.

Tabellen sind in Latex sehr kompliziert zu erzeugen. Alternativ kann man die Tabellen auch in einem anderen Programm gestalten und als Bild wieder einfügen. Dieses Bild kann dann innerhalb von begin-Table verwendet werden.

## 11.2. Beispiele Listen

Im Folgenden wird eine Liste gezeigt:

- Ich weiß, dass viele Geräte des täglichen Lebens durch Computer gesteuert werden und kann für mich relevante nennen und nutzen.

## Prototyp: NeXt

DW OR N PACKAGE (TOP VIEW)	
NC	1 20 NC
V <sub>CC</sub>	2 19 GND
SER IN	3 18 SER OUT
DRAIN0	4 17 DRAIN7
DRAIN1	5 16 DRAIN6
DRAIN2	6 15 DRAIN5
DRAIN3	7 14 DRAIN4
SRCLR	8 13 SRCK
G	9 12 RCK
GND	10 11 GND

NC – No internal connection

$V_{cc}$  Positive supply voltage  
 GND Ground  
 SER IN Daten Pin  
 SRCK Clock Pin  
 RCK Latch Pin  
 $\overline{SRCLR}$  Wenn **shift-register clear** LOW ist,  
werden die input Register gelöscht  
 $\overline{G}$  Wenn **output enable** HIGH ist, werden  
die Daten im Output Buffer LOW gehal-  
ten

Tabelle 11.1.: Aufwändige Tabelle mit Abbildung und Caption

1. Und jetzt eine Numerierung
  2. Und jetzt eine Numerierung
- Ich kann wichtige Bestandteile eines Computersystems (Eingabe-, Ausgabe-geräte und Zentraleinheit) benennen, kann ihre Funktionen beschreiben und diese bedienen.

Und jetzt eine Numerierung:

1. Aufzählungspunkt
  - a) Unteraufzählung
  - b) Unteraufzählung
    - Und jetzt noch eine Ebene ohne Aufzählung
    - Und jetzt noch eine Ebene ohne Aufzählung
2. Aufzählungspunkt
3. Aufzählungspunkt
4. Aufzählungspunkt
5. Aufzählungspunkt

## 11.3. Beispiel Codesequenz

In Listing 11.1 sieht man ein Quick-Sort-Listing in der Programmiersprache JAVA. Das Listings-Paket übernimmt die Formatierung von Codebausteinen und kann in der Präambel nach Belieben auf eine andere Sprache konfiguriert werden.

### 11.3.1. Quicksort in JAVA

Quelltext 11.1: QuickSort in Java

```
1 public class QuickSort
2 {
3     public static void main(String[] args)
4     {
5         int [] x =
6         {
7             9, 2, 4, 7, 3, 7, 10
8         }
9         ;
10        System.out.println(Arrays.toString(x));
11
12        int low = 0;
13        int high = x.length - 1;
14
15        quickSort(x, low, high);
16        System.out.println(Arrays.toString(x));
17    }
18
19    public static void quickSort(int[] arr, int low,
20                                int high)
21    {
22        if (arr == null || arr.length == 0)
23            return;
```

```
24     if (low >= high)
25         return;
26
27     // pick the pivot
28     int middle = low + (high - low) / 2;
29     int pivot = arr[middle];
30
31     // make left < pivot and right > pivot
32     int i = low, j = high;
33     while (i <= j)
34     {
35         while (arr[i] < pivot)
36         {
37             i++;
38         }
39
40         while (arr[j] > pivot)
41         {
42             j--;
43         }
44
45         if (i <= j)
46         {
47             int temp = arr[i];
48             arr[i] = arr[j];
49             arr[j] = temp;
50             i++;
51             j--;
52         }
53     }
54
55     // recursively sort two sub parts
56     if (low < j)
57         quickSort(arr, low, j);
58 }
```

```
59     if (high > i)
60         quickSort(arr, i, high);
61     }
62 }
```

## 11.4. Beispieltext

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Das hier ist der zweite Absatz. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Und nun folgt – ob man es glaubt oder nicht – der dritte Absatz. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleich-

gültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Nach diesem vierten Absatz beginnen wir eine neue Zählung. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

# Literaturverzeichnis

[Schwab 2013] SCHWAB, Felix: *Systemplanung und Projektentwicklung - Projektmanagement und Software - Entwicklung*. Wien : Manz Verlag Schulbuch, 2013. – ISBN 978-3-706-84352-2

[Wikipedia 2018a] WIKIPEDIA: *LaTeX — Wikipedia, Die freie Enzyklopädie*. 2018. – URL <https://de.wikipedia.org/w/index.php?title=LaTeX&oldid=178471003>. – [Online; Stand 23. Juni 2018]

[Wikipedia 2018b] WIKIPEDIA: *Spiel-Engine — Wikipedia, Die freie Enzyklopädie*. 2018. – URL <https://de.wikipedia.org/w/index.php?title=Spiel-Engine&oldid=173243484>. – [Online; Stand 22. Juni 2018]

[Wikipedia 2018c] WIKIPEDIA: *Unity (Spiel-Engine) — Wikipedia, Die freie Enzyklopädie*. 2018. – URL [https://de.wikipedia.org/w/index.php?title=Unity\\_\(Spiel-Engine\)&oldid=178184986](https://de.wikipedia.org/w/index.php?title=Unity_(Spiel-Engine)&oldid=178184986). – [Online; Stand 23. Juni 2018]

# **A. Anhang-Kapitel**

## **A.1. Anhang-Section**

Testtext