

Logo  
Projektpartner



## Diplomarbeit

# Prototyp: NeXt

**Untertitel der Arbeit**

Michael Leitner      Lukas Vogel

Imst, 22. 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)

Tabelle 0.1.: 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 ein Prototyp für das Computerspiel NeXt, welches Elemente aus dem Puzzle und 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.

# **Abstract (Englisch)**

(ca. ½ bis max. 2 Seiten)

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>11</b>
<b>Tabellenverzeichnis</b>	<b>12</b>
<b>Quelltexte</b>	<b>13</b>
<b>1. Einleitung</b>	<b>15</b>
<b>2. Projektmanagement</b>	<b>16</b>
2.1. Metainformationen . . . . .	16
2.1.1. Team . . . . .	16
2.1.2. Betreuer . . . . .	17
2.1.3. Partner . . . . .	18
2.1.4. Ansprechpartner . . . . .	18
2.2. Vorerhebungen . . . . .	18
2.2.1. Projektzieleplan . . . . .	18
2.2.2. Projektumfeld . . . . .	19
2.2.3. Risikoanalyse . . . . .	19
2.3. Pflichtenheft . . . . .	19
2.3.1. Zielbestimmung . . . . .	19
2.3.2. Produkteinsatz und Umgebung . . . . .	19
2.3.3. Funktionalitäten . . . . .	20
2.3.4. Testszenarien und Testfälle . . . . .	20
2.3.5. Liefervereinbarung . . . . .	20
2.4. Planung . . . . .	21
2.4.1. Projektstrukturplan . . . . .	21
2.4.2. Meilensteine . . . . .	21
2.4.3. Gant-Chart . . . . .	21

2.4.4. Abnahmekriterien . . . . .	21
2.4.5. Pläne zur Evaluierung . . . . .	21
2.4.6. Ergänzungen und zu klärende Punkte . . . . .	21
<b>3. Vorstellung des Produktes</b>	<b>22</b>
3.1. Produktbeschreibung . . . . .	22
<b>4. Eingesetzte Technologien</b>	<b>23</b>
4.1. Unity-Engine . . . . .	23
4.1.1. Spiele-Engine-Erklärung . . . . .	23
<b>5. Problemanalyse</b>	<b>24</b>
5.1. USE-Case-Analyse . . . . .	24
5.2. Domain-Class-Modelling . . . . .	25
5.3. User-Interface-Design . . . . .	25
<b>6. Systementwurf</b>	<b>26</b>
6.1. Architektur . . . . .	26
6.1.1. Design der Komponenten . . . . .	26
6.1.2. Benutzerschnittstellen . . . . .	27
6.1.3. Datenhaltungskonzept . . . . .	27
6.1.4. Konzept für Ausnahmebehandlung . . . . .	27
6.1.5. Sicherheitskonzept . . . . .	27
6.1.6. Design der Testumgebung . . . . .	27
6.1.7. Design der Ausführungsumgebung . . . . .	28
6.2. Detailentwurf . . . . .	28
<b>7. Implementierung</b>	<b>30</b>
<b>8. Deployment</b>	<b>31</b>
<b>9. Tests</b>	<b>32</b>
9.1. Systemtests . . . . .	32
9.1.1. Einführung . . . . .	32
9.1.2. Testfälle . . . . .	32
9.2. Akzeptanztests . . . . .	32
<b>10. Projektevaluation</b>	<b>37</b>

<b>11. Benutzerhandbuch</b>	<b>38</b>
<b>12. Betriebswirtschaftlicher Kontext</b>	<b>39</b>
<b>13. Zusammenfassung</b>	<b>40</b>
13.1. Projektevaluation . . . . .	40
13.2. Produktevaluierung . . . . .	41
13.3. Resümee . . . . .	42
13.3.1. Resümee (Leitner Michael) . . . . .	42
13.3.2. Resümee (Vogel Lukas) . . . . .	42
<b>14. Beispielkapitel</b>	<b>44</b>
14.1. Beispiele zitieren . . . . .	44
14.1.1. Beispiele Abbildungen . . . . .	44
14.2. Beispiele Listen . . . . .	45
14.3. Beispiel Codesequenz . . . . .	47
14.3.1. Quicksort in JAVA . . . . .	47
14.4. Beispieltext . . . . .	49
<b>Literaturverzeichnis</b>	<b>51</b>
<b>A. Anhang-Kapitel</b>	<b>52</b>
A.1. Anhang-Section . . . . .	52

# **Abbildungsverzeichnis**

2.1.	Mag. Claudio Landerer . . . . .	17
2.2.	ClockStone Softwareentwicklung GmbH . . . . .	18
2.3.	Mag. Claudio Landerer . . . . .	18
14.1.	Arduino mit Lichtsensor und Lichterkette . . . . .	45

# **Tabellenverzeichnis**

0.1.	Abstract Deutsch . . . . .	6
2.1.	Team . . . . .	16
2.2.	Leitner . . . . .	16
2.3.	Vogel . . . . .	16
9.1.	Testfall 1: Bewegungssteuerung . . . . .	33
9.2.	Testfall 2: Levelabschluss . . . . .	34
9.3.	Testfall 3: Tod der Spielfigur durch Fall . . . . .	35
9.4.	Testfall 4: Schalter zum Öffnen einer Tür . . . . .	36
14.1.	Aufwändige Tabelle mit Abbildung und Caption . . . . .	46

# **Quelltexte**

14.1. QuickSort in Java . . . . . 47

# **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

Tabelle 2.1.: Team

 A professional headshot of a young man with dark hair and glasses, wearing a dark suit jacket over a grey shirt.	 A photograph of a young man with light brown hair, wearing a white Puma t-shirt with the number 14 and the Austrian national football team logo on the shoulder.
Tabelle 2.2.: Leitner	Tabelle 2.3.: Vogel
Team-Leiter	Teammitglied
Lukas Vogel	Leitner Michael

### **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 2.1.: Mag. Claudio Landerer

*Prototyp: NeXt*

### **2.1.3. Partner**



Abbildung 2.2.: ClockStone Softwareentwicklung GmbH

Unser Partner während der Diplomarbeit war die Firma ClockStone Softwareentwicklung GmbH. Unser Ansprechpartner war [NAME].

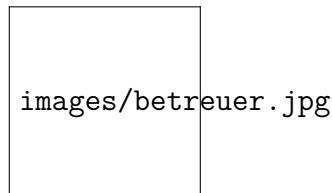


Abbildung 2.3.: Mag. Claudio Landerer

### **2.1.4. Ansprechpartner**

## **2.2. Vorerhebungen**

### **2.2.1. Projektzieleplan**

Projektziele-Hierarchie - SMART

## **2.2.2. Projektumfeld**

- Identifikation der Stakeholder
- Charakterisierung der Stakeholder
- Maßnahmen
- Grafische Darstellung des Umfeldes

## **2.2.3. Risikoanalyse**

- Risikomatrix

# **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
- ...

### **2.3.5. Liefervereinbarung**

- Lieferumfang
- Modus
- Verteilung(Deployment)

## **2.4. Planung**

### **2.4.1. Projektstrukturplan**

### **2.4.2. Meilensteine**

### **2.4.3. Gant-Chart**

### **2.4.4. Abnahmekriterien**

### **2.4.5. Pläne zur Evaluierung**

### **2.4.6. 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**

- Kurzbeschreibung aller Technologien, die verwendet wurden.
- Technologien die aus dem Unterricht bekannt sind, nur nennen und deren Einsatzzweck im Projekt beschreiben, nicht die Technologien selbst.
- Technologien die aus dem Unterricht nicht bekannt sind, im Detail beschreiben incl. deren Einsatz im Projekt
- Fokus aus eingesetzten Frameworks

### **4.1. Unity-Engine**

#### **4.1.1. Spiele-Engine-Erklärung**

Eine Engine ist eine Art Framework

# **5. Problemanalyse**

## **5.1. USE-Case-Analyse**

- UseCases auf Basis von Benutzerzielen identifizieren:
  - Benutzer eines Systems identifizieren
  - Benutzerziele identifizieren (Interviews)
  - Use-Case-Liste pro Benutzer definieren
- UseCases auf Basis von Ereignissen identifizieren:
  - Externes Event triggert einen Prozess
  - zeitliches Event triggert einen Prozess (Zeitpunkt wird erreicht)
  - State-Event (Zustandsänderung im System triggert einen Prozess)
- Werkzeuge:
  - USE-Case-Beschreibungen (textuell, tabellarisch)
  - USE-Case-Diagramm
  - Aktivitätsdiagramm für den Use-Case (Interaktion zwischen Akteur und System abbilden)
  - System-Sequenzdiagramm (Spezialfall eines Sequenzdiagramms: Nur 1 Akteur und 1 Objekt, das Objekt ist das komplette System, es geht um die Input/Output Requirements, die abzubilden sind)

## **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**

- Mockups
- Wireframes

# **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

Detaillierte Beschreibung der Implementierung aller Teilkomponenten der Software entlang der zentralsten Use-Cases:

- 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. Deployment**

- Umsetzung der Ausführungsumgebung
- Deployment
- DevOps-Thema

# **9. Tests**

## **9.1. Systemtests**

### **9.1.1. Einführung**

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

### **9.1.2. Testfälle**

## **9.2. Akzeptanztests**

Tabelle 9.1.: Testfall 1: Bewegungssteuerung

<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 9.2.: Testfall 2: Levelabschluss

<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 anderes 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 9.3.: Testfall 3: Tod der Spielfigur durch Fall

<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 9.4.: Testfall 4: Schalter zum Öffnen einer Tür

<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>

# **10. Projektevaluation**

siehe Projektmanagement-Unterricht

# **11. Benutzerhandbuch**

falls im Projekt gefordert

# **12. Betriebswirtschaftlicher Kontext**

BW-Teil

# **13. Zusammenfassung**

## **13.1. Projektevaluation**

Während des Projektes lernten wir schnell das wir den Arbeitsaufwand des Projektes überschätzt 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 zu dem normalen Schultag nötig.

Die Zusammenarbeit des Teams war nie ein Problem, dass einzige was hierbei ein Rückschlag war, war das ausfallen eines Teammitglieds und die dadurch entstandene zusätzliche Arbeit. Da wir durch diesen Ausfall die Verteilung der Themen sowie die Organisation als Projektteam nochmals neu überdenken mussten.

Auch bei der Zuordnung der Themen traten keine Probleme auf, da jeder im Team seinen wunschschwerpunkt ohne Diskussion bekam. Dies ist darauf zurück zu führen, dass wir diesbezüglich unterschiedlich sind und es bei diesem Projekt genau passte. Die größte Schwierigkeit war das Zeitmanagement, da das gesamte Team nicht nur die Diplomarbeit bearbeiten musste, sondern zusätzlich noch die benötigten schulischen Leistungen erbringen musste. Dies führte des Öfteren zu Komplikationen und somit auch zu zeitlichen Verschiebungen von Meilensteinen.

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.

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

## **13.2. Produktevaluierung**

### 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.

## 13.3. Resümee

### 13.3.1. Resümee (Leitner Michael)

Die Entwicklung eines Spiels 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.“

### 13.3.2. 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

## *Prototyp: NeXt*

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 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.

# **14. Beispielkapitel**

## **14.1. Beispiele zitieren**

Wikipedia (2018)

### **14.1.1. Beispiele Abbildungen**

Auf diese Weise kann man zum Beispiel in Latex auf die Abbildung 14.1 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 14.1 auf Seite 45 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

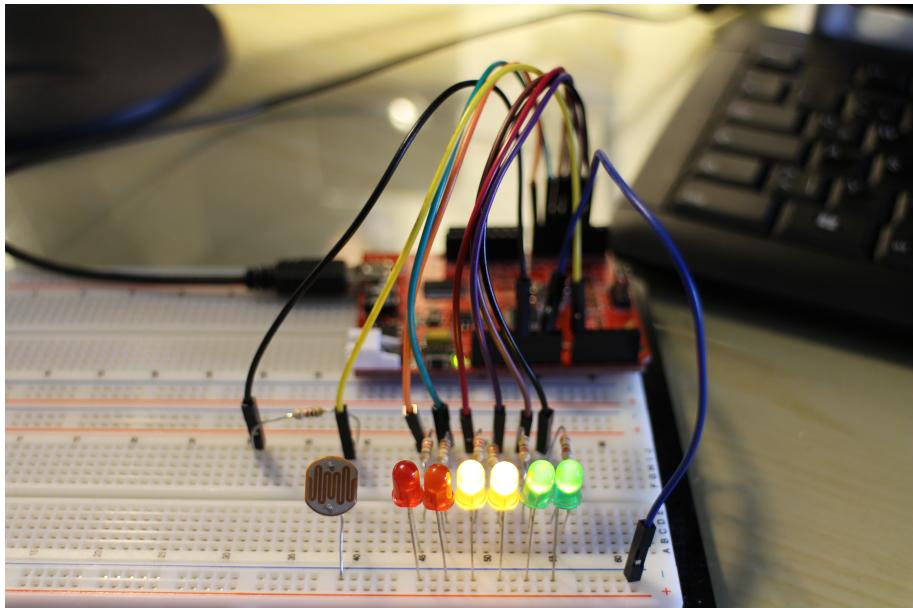


Abbildung 14.1.: Hintergrund: Arduino Board; Vordergrund: eine Lichterreihe und ein Lichtsensor (Fotowiderstand); In diesem Beispiel wird die Lichterreihe je nach Helligkeit des Umgebungslichtes gesteuert. Durch leichte Modifikationen kann man damit eine Lichtschranke oder auch eine Helligkeitssteuerung für das Smartphone simulieren.

## Beispiele Tabellen

Tabelle 14.1 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.

## 14.2. 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
$V_{CC}$	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
$\overline{SRCLR}$	8	13 SRCK
$\overline{G}$	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 <b>shift-register clear</b> LOW ist, werden die input Register gelöscht
	$\overline{G}$	Wenn <b>output enable</b> HIGH ist, werden die Daten im Output Buffer LOW gehalten

Tabelle 14.1.: 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

## 14.3. Beispiel Codesequenz

In Listing 14.1 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.

### 14.3.1. Quicksort in JAVA

Quelltext 14.1: 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 }
```

## 14.4. 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

[Wikipedia 2018] 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]

# **A. Anhang-Kapitel**

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

Testtext