

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

z. B. Hinweise, wie das bearbeitete Thema gefunden wurde oder Dank für die Betreuung (Kooperationspartner/in, Betreuer/innen, Sponsoren) etc.

Abstract (Deutsch)

Das Ziel der vorliegenden Diplomarbeit war es, ein Computerspiel des Jump and Run und des Puzzle Genres zu entwickeln. Für diesen Zweck wurde die Unity-Engine verwendet und die vom selben Hersteller mitgelieferte Entwicklungsumgebung welche auf der Programmiersprache C# basiert. In der Diplomarbeit werden wesentlichen Source-Code Elemente aus dem Spiel veranschaulicht und auch erklärt und somit die Entwicklung des Projekts gezeigt. Diese Dokumentation bietet einen Einblick in die Spielentwicklung und kann somit für jeden eine Hilfe sein der sich mit diesem Thema beschäftigt.

Tabelle 0.1.: Abstract Deutsch

Thema:	Prototyp: NeXt
Name der Verfasser:	Michael Leitner & Lukas Vogel
Jahrgang:	—
Schuljahr:	2017/18
Kooperationspartner:	ClockStone Softwareentwicklung GmbH

Aufgabenstellung | —

Realisierung: | —

Ergebnisse: | —

Abstract (Englisch)

(ca. ½ bis max. 2 Seiten)

Inhaltsverzeichnis

Abbildungsverzeichnis	10
Tabellenverzeichnis	11
Quelltexte	12
1. Einleitung	15
2. Projektmanagement	16
2.1. Metainformationen	16
2.1.1. Team	16
2.1.2. Betreuer	16
2.1.3. Partner	16
2.1.4. Ansprechpartner	16
2.2. Vorerhebungen	16
2.2.1. Projektzieleplan	16
2.2.2. Projektumfeld	16
2.2.3. Risikoanalyse	17
2.3. Pflichtenheft	17
2.3.1. Zielbestimmung	17
2.3.2. Produkteinsatz und Umgebung	17
2.3.3. Funktionalitäten	18
2.3.4. Testszenarien und Testfälle	18
2.3.5. Liefervereinbarung	18
2.4. Planung	19
2.4.1. Projektstrukturplan	19
2.4.2. Meilensteine	19
2.4.3. Gant-Chart	19

2.4.4. Abnahmekriterien	19
2.4.5. Pläne zur Evaluierung	19
2.4.6. Ergänzungen und zu klärende Punkte	19
3. Vorstellung des Produktes	20
3.1. Produktbeschreibung	20
4. Eingesetzte Technologien	21
4.1. Unity-Engine	21
4.1.1. Spiele-Engine-Erklärung	21
5. Problemanalyse	22
5.1. USE-Case-Analyse	22
5.2. Domain-Class-Modelling	23
5.3. User-Interface-Design	23
6. Systementwurf	24
6.1. Architektur	24
6.1.1. Design der Komponenten	24
6.1.2. Benutzerschnittstellen	25
6.1.3. Datenhaltungskonzept	25
6.1.4. Konzept für Ausnahmebehandlung	25
6.1.5. Sicherheitskonzept	25
6.1.6. Design der Testumgebung	25
6.1.7. Design der Ausführungsumgebung	26
6.2. Detailentwurf	26
7. Implementierung	28
8. Deployment	29
9. Tests	30
9.1. Systemtests	30
9.1.1. Einführung	30
9.1.2. Testfälle	30
9.2. Akzeptanztests	30
10. Projektevaluation	35

11. Benutzerhandbuch	36
12. Betriebswirtschaftlicher Kontext	37
13. Zusammenfassung	38
13.1. Projektevaluation	38
13.2. Produktevaluierung	39
13.3. Resümee	40
13.3.1. Resümee (Leitner Michael)	40
13.3.2. Resümee (Vogel Lukas)	40
14. Beispielkapitel	42
14.1. Beispiele zitieren	42
14.1.1. Beispiele Abbildungen	43
14.2. Beispiele Listen	44
14.3. Beispiel Codesequenz	46
14.3.1. Quicksort in JAVA	46
14.4. Beispieltext	48
Literaturverzeichnis	50
A. Anhang-Kapitel	52
A.1. Anhang-Section	52

Abbildungsverzeichnis

14.1. Arduino mit Lichtsensor und Lichterkette	44
----------------------------------------------------------	----

Tabellenverzeichnis

0.1. Abstract Deutsch	5
9.1. Testfall 1: Bewegungssteuerung	31
9.2. Testfall 2: Levelabschluss	32
9.3. Testfall 3: Tod der Spielfigur durch Fall	33
9.4. Testfall 4: Schalter zum Öffnen einer Tür	34
14.1. Aufwändige Tabelle mit Abbildung und Caption	45

Quelltexte

14.1. QuickSort in Java 46

Einleitende Bemerkungen

Notationen

Beschreibung wie Code, Hinweise, Zitate etc. formatiert werden

1. Einleitung

Diese Diplomarbeit handelt von der Entwicklung des Prototyps: NeXt. Das Produkt ist eine spielbare Version eines Computerspiels in dem sogenannte "Jump and RunElemente, Puzzle-Elemente sowie Zeitmanagement-Elemente beinhaltet sind. Das Projekt wurde in Verbindung mit der Firma ColockStone Softwareentwicklung GmbH bearbeitet. 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 da gelegt. Wir hoffen, dass dieses Dokument für diese Personen hilfreich ist. Außerdem möchten wir uns ganz herzlich bei unseren Betreuern seitens der Schule, Herr Claudio Landerer und seitens der Firma. Sie unterstützen uns immer wenn wir Fragen hatten und motivierten uns falls nötig. Die Themen welche Sie hier lesen können sollten alle verständlich erklärt sein.

2. Projektmanagement

2.1. Metainformationen

2.1.1. Team

2.1.2. Betreuer

2.1.3. Partner

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

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

Tabelle 9.2.: Testfall 2: Levelabschluss

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

Tabelle 9.3.: Testfall 3: Tod der Spielfigur durch Fall

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

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

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

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 wird 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

Das ist ein Zitat mit Klammern, (Resnick, 1996), das ein Zitat ohne Klammern: Harel und Papert (1991). Hier das selbe Zitat mit einer Seitenangabe und Klammern (Resnick, 1996, S. 23).

Wird ein Absatz aus einer Quelle sinngemäß übernommen (nicht wörtlich), dann kann nach dem Absatz das entsprechende Zitat in Klammern angeführt werden. (Anastopoulou u.a., 2012, S. 33)

Wenn ein Zitat im Text angegeben wird, wie z.B. so Beer, Rudolf und Benischek, Isabella (2011), können die Klammern weggelassen werden.

Der folgende Absatz zeigt ein Blockzitat (wörtlich übernommene Textpassage aus einer Quelle):

Dr. Heinrich Faust ist ein angesehener Wissenschaftler und Akademiker, der trotz seiner wissenschaftlichen Studien und einer guten Bildung seinen Wissensdurst nicht stillen kann. Eines Nachts sitzt er in seinem Studierzimmer und grübelt über den Sinn des Lebens nach, findet jedoch keine Antworten. Daraufhin wendet er sich der Geisterwelt zu. Er beschwört einen Erdgeist, versucht sich den Geistern gleich zu stellen, was ihm jedoch nicht gelingt. Von Ohnmacht getrieben will er sich das Leben nehmen. Sein Selbstmordversuch wird jedoch von Glockenläuten zum Ostertag und seinen Kindheitserinnerungen gestört. (Ackermann,

Prototyp: NeXt

2001, S. 21)

Hier wird ein wörtliches Zitat inline angegeben: „Das ist ein kleines direktes Zitat.“ Göhlich und Zirfas (2007), und danach geht es gleich wieder direkt weiter. Ob ein wörtliches Zitat inline oder als eigener Block angezeigt wird, entscheidet Latex auf Basis der Länge.

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.

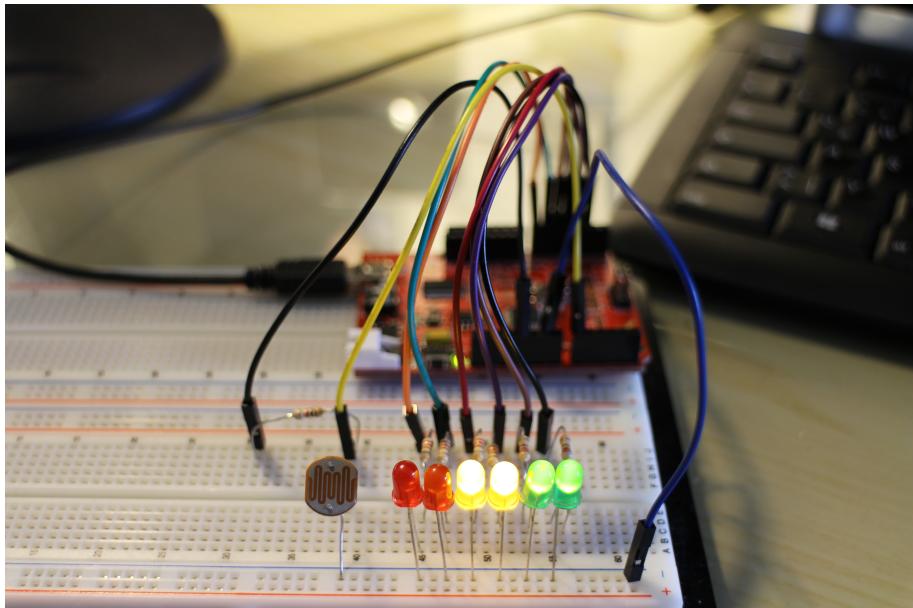
Der nächste wichtige Punkt sind die Quellenangaben bei Abbildungen. Der Author muss zu jeder Abbildung die notwendigen Rechte haben und idealer Weise gibt man diese bei der Abbildung mit an. In Abbildung 14.1 auf Seite 44 sieht man das.

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

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 Tabel-



© Stefan Stolz (CC BY-SA 3.0)

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.

len 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.
 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.

Prototyp: NeXt

DW OR N PACKAGE (TOP VIEW)	
NC	1 20 NC
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
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 14.1.: Aufwändige Tabelle mit Abbildung und Caption

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

- [Ackermann 2001] ACKERMANN, Edith: Piaget's constructivism, Papert's constructionism: What's the difference. In: *Future of learning group publication* 5 (2001), Nr. 3, S. 438. – URL http://lovettresourcenetwork.wiki.lovett.org/file/view/EA.Piaget+_+Papert.pdf. – Zugriffssdatum: 2014-07-09
- [Anastopoulou u. a. 2012] ANASTOPOULOU, Stamatina ; BERLAND, Matthew ; FRANT, Janete B. ; BOYTCHEV, Pavel ; BRENNAN, Karen ; CHRONAKI, Anna ; CLAYSON, James ; CORREIA, Secundino ; DAGIENE, Valentina ; DEKOLI, Margarita: Constructionism 2012 Theory Practice and Impact. (2012), August. – URL http://users.uoa.gr/~zsmyrnaiou/conferences_after2008/constructionism%201_2012.pdf. – Zugriffssdatum: 2014-03-26
- [Beer, Rudolf und Benischek, Isabella 2011] BEER, RUDOLF ; BENISCHEK, ISABELLA: Aspekte kompetenzorientierten Lernens und Lehrens. In: BIFIE (Hrsg.): *Kompetenzorientierter Unterricht in Theorie und Praxis*. Graz : Leykam, 2011
- [Göhlich und Zirfas 2007] GÖHLICH, Michael ; ZIRFAS, Jörg: *Lernen: Ein pädagogischer Grundbegriff*. Stuttgart : Kohlhammer, April 2007. – ISBN 9783170188693
- [Harel und Papert 1991] HAREL, Idit ; PAPERT, Seymour: *Situating Constructionism*. Norwood, N.J : Ablex Publishing Corporation, U.S., 1991. – ISBN 9780893917869
- [Resnick 1996] RESNICK, Mitchel: Distributed constructionism. In: *Proceedings of the 1996 international conference on Learning sciences*, International Society of

Prototyp: NeXt

the Learning Sciences, 1996, S. 280–284. – URL <http://dl.acm.org/citation.cfm?id=1161173>. – Zugriffsdatum: 2015-04-20

A. Anhang-Kapitel

A.1. Anhang-Section

Testtext