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 consist 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

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltexte

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, Puzzele-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
- Leveldesign
- Dokumentation

Abbildung 2.2.: Michael Leitner

Name: Micahel 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 unterstütze 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

Das Ziel unseres Projekts ist es ein spielbares, unterhaltendes und einfach bedienbares Computer Spiel zu entwickeln. Die Meilensteine sollen Termingerecht realisiert werden und es soll pünktlich bis zur Projektabgabe im Juni fertig gestellt werden.

2.2.2. Projektumfeld

Stakeholder-Analyse

Die Ergebnisse der Stakholder-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

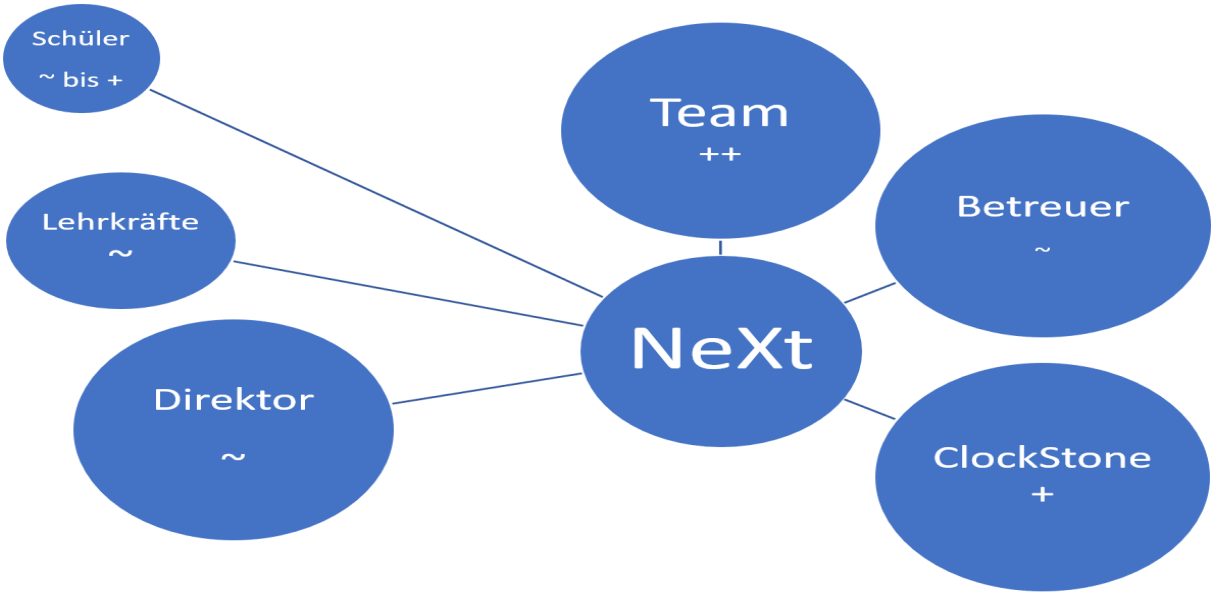


Abbildung 2.6.: Stakeholder-Diagramm

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

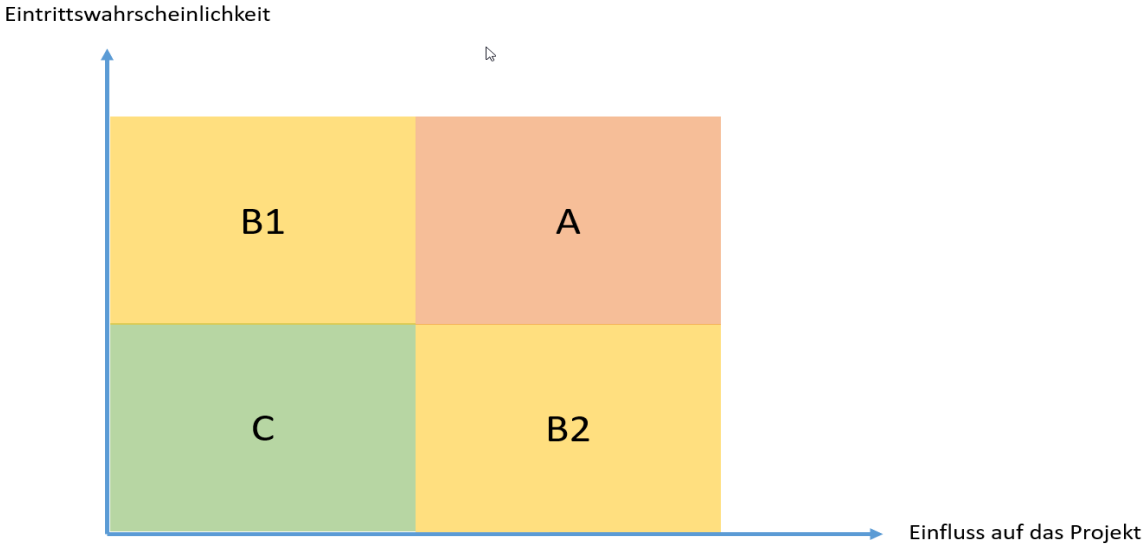


Abbildung 2.7.: Matrix zur qualitativen Einordnung von Risiken

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 Erfolgserlebnisse 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 bitte.
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	• laufende Berichterstattung • laufender Fortschritt im Projekt • bei Problemen oder Verzögerungen mit ihm reden
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

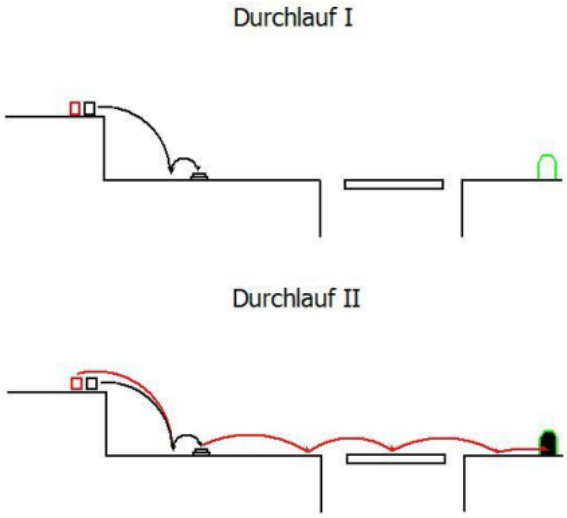
2.3. Pflichtenheft

2.3.1. Zielbestimmung

Das Spiel „NeXt“ soll nach dem „Time-Rifts“-Prinzip agieren. Das 2-Dimensionale Spiel wird gemeistert durch:

- koordinierte Zusammenarbeit aller Figuren

- Zeitmanagement
- Jump-and-Run Elemente
- Lösen von Puzzle-Elementen



Time-Rifts (Erklärung): Jeder Durchlauf hat ein Zeitlimit, dass Ziel sollte es sein Durch die verschlossene Tür (grün) zu gehen.

- Durchlauf I: Der/die Spieler/in bewegt Figur 1(schwarzes Rechteck) auf eine Bodenplatte. -> Tür öffnet sich. (bleibt offen solange die Bodenplatte betätigt ist)
- Durchlauf II: Figur 1 macht das selbe wieder (in Echtzeit) und der/die Spieler/in bewegt Figur 2 (rotes Rechteck) durch die Tür

Jedoch ist dieses Prinzip nicht auf nur 2 Durchgänge beschränkt.

2.3.2. Produkteinsatz und Umgebung

Die Zielgruppe dieses Spieles, ist jeder der sich einer Herausforderung stellen möchte. Durch das TimeRift-Prinzip wird eine großer Herausforderung geboten.Spielbar wird dieses Spiel auf jedem Windows-Computer sein. Da auf gute Grafik verzichtet worden ist, muss der Computer auch nicht leistungsstark sein, um das Spiel ausführen zu können

2.3.3. Funktionalitäten

- MUSS-Anforderungen
  - Funktional:
    - TimeRift-Prinzip-Implementiert
    - Ein paar Level implementiert
  - Nicht-funktional
    - Das Spiel soll einfach gesteuert werden
    - Das Spiel soll Einschränkungen Spielbar sein.
- KANN-Anforderungen
  - Funktional
    - Ein Komplettes Level-Pack enthalten
    - Erweiterte Steuerungsmöglichkeiten
  - Nicht-funktional
    - Das Spiel kann mit Musik unterlegt sein

2.4. Planung

2.4.1. Projektstruktrplan

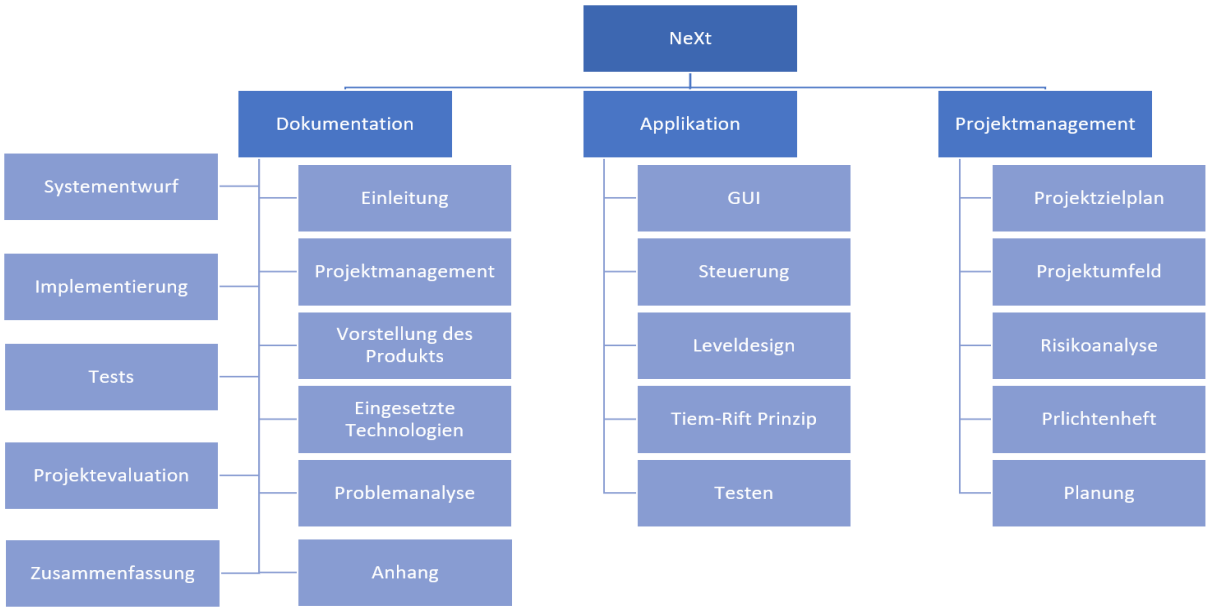


Abbildung 2.8.: Projektstrukturplan

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

2.4.2. Meilensteine

- 10.10.2017 Abschluss Anforderungsdefinition, Vorbereitung der Entwicklungsumgebung
- 28.11.2017 Erster spielbarer Prototyp (Basissteuerung, Physik, Sound, Grafik, Test-Level)
- 09.01.2018 Spielbarer Prototyp nach dem Timerift-Prinzip
- 06.02.2018 Erste spielbare Levels

- 20.03.2018 Vollständige Fertigstellung des Spiels mit einigen Levels, Beginn der Beta-Testing-Phase
- 06.04.2018 Abschluss Beta-Testing
- 30.04.2018 Verbesserung des Prototyps auf Basis der Rückmeldungen (Beta-Test)
- 15.06.2018 Fertigstellung der Dokumentation, Übergabe an den Auftraggeber

2.4.3. Gant-Chart

Vorgang	Anfang	Ende
Projektbeginn	06.11.17	06.11.17
Dokumentation	06.11.17	29.06.18
Pflichtenheft	06.11.17	10.11.17
Entwurfsphase	10.11.17	24.11.17
Literaturrecherche	27.11.17	18.12.17
Leveldesign	27.11.17	06.02.18
Steuerung	25.12.17	31.01.18
TimeRit-Prinzip	01.02.18	30.03.18
GUI	08.01.18	07.02.18
Hauptmenü	08.01.18	11.01.18
Levelmenü	12.01.18	17.01.18
Pausenmenü	22.01.18	26.01.18
Verfeinerung	02.04.18	29.06.18
Projektabschluss	29.06.18	29.06.18

Abbildung 2.9.: Vorgänge des Gantt Charts

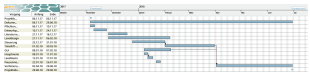


Abbildung 2.10.: Gantt Chart

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 Puzzele-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 Technische 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 könne 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 Apllikatione 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

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.

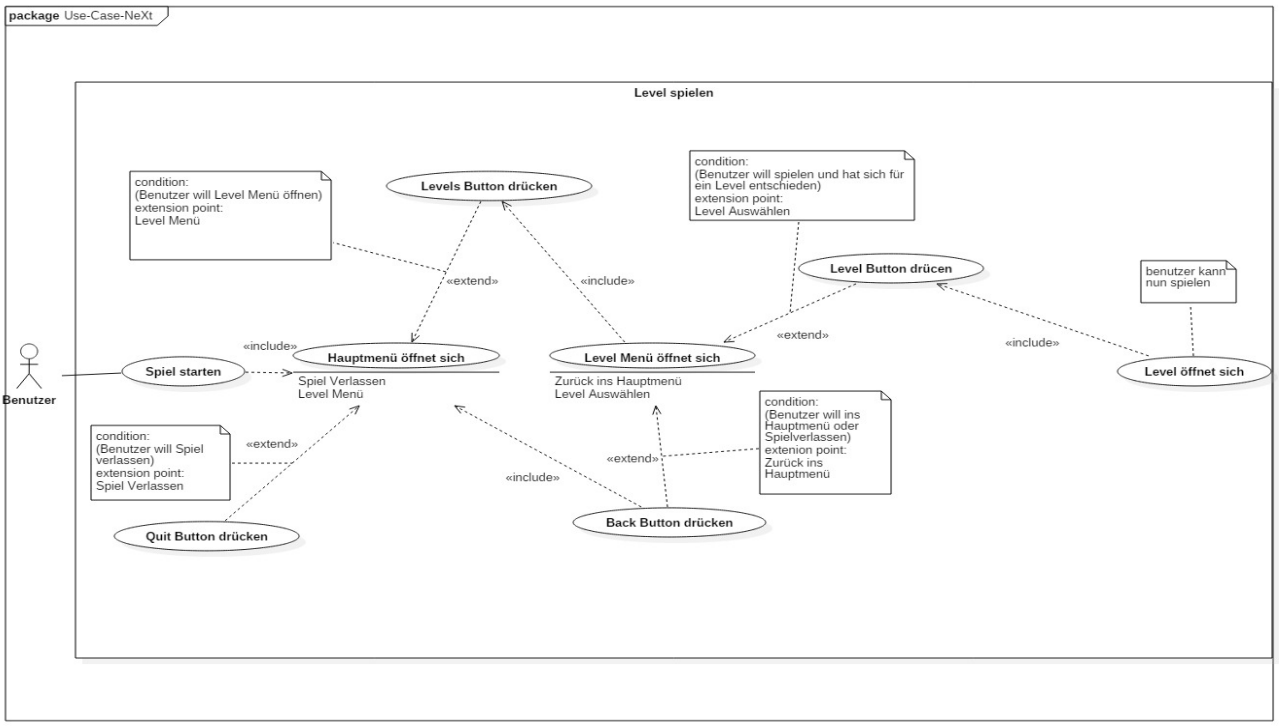


Abbildung 5.1.: USE-Case-Diagramm

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.

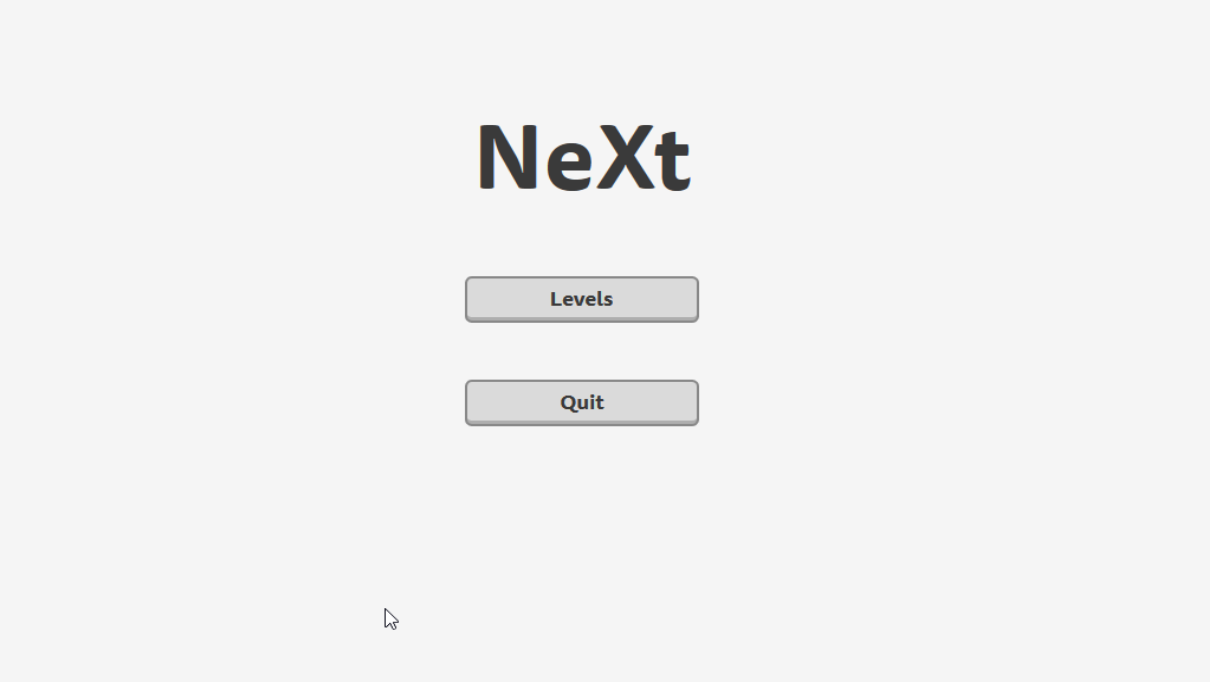


Abbildung 5.2.: Wireframe-Hauptmenü

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.

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.

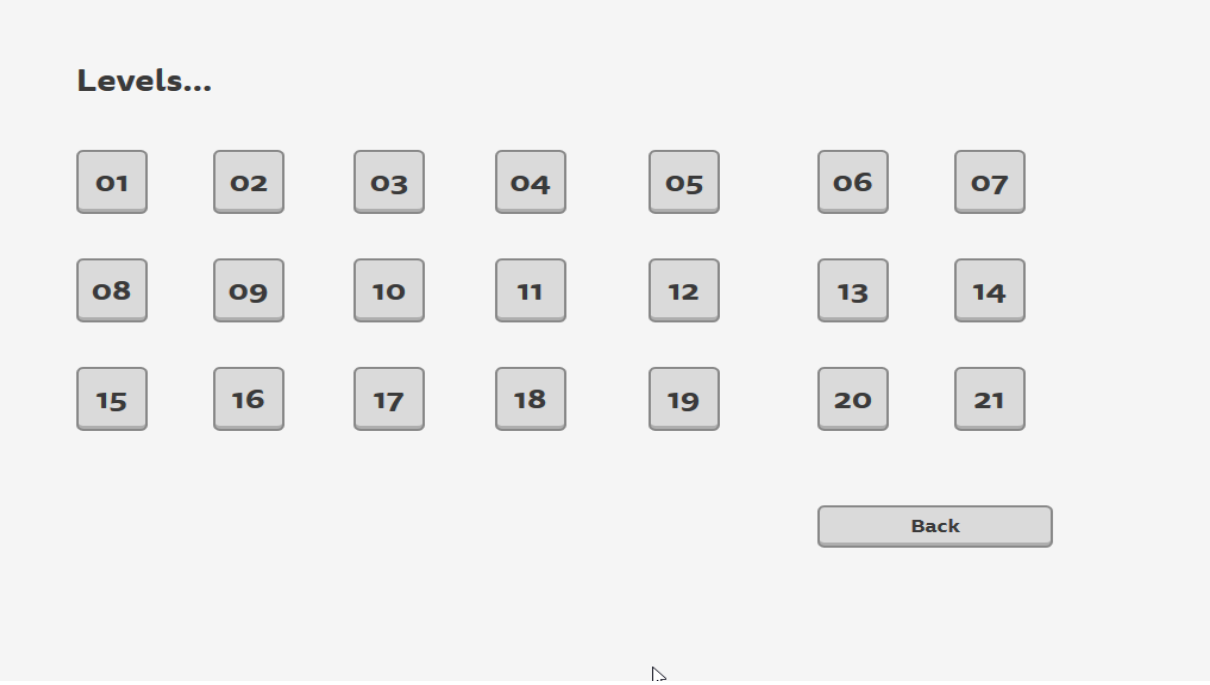


Abbildung 5.3.: Wireframe-Levelmenü

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.

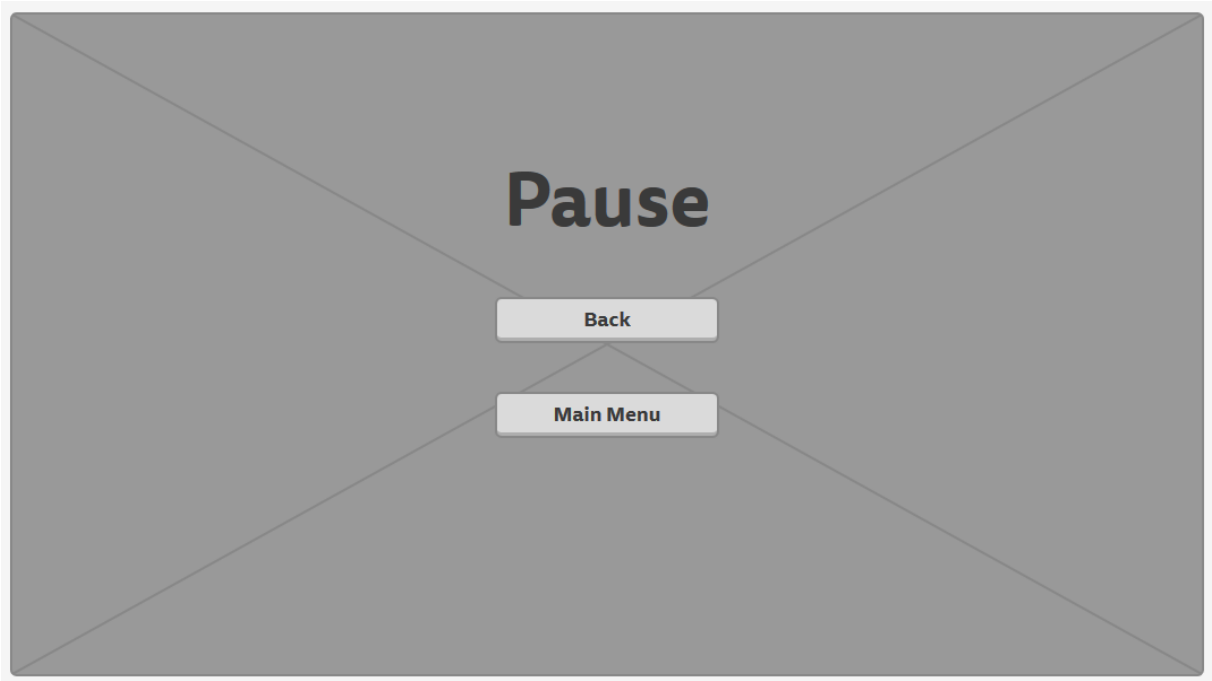


Abbildung 5.4.: Wireframe-Pausemenü

Sobald der Benutzer sich in einem Level befindet soll er die Möglichkeit habben, 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 denn Benutzer wieder in die Ansicht des Hauptmenüs, Abbildung 5.2 bringen.

6. Systementwurf

6.1. Architektur

6.1.1. Design der Komponenten

In der Unity3D Entwicklungsumgebung werden keine Klassischen Architekturmuster verwendet. Wenn ein bestimmtes Objekt ein Skript benötigt, wird ihm dieses Skript angehängt. Jedoch wurden für das TimeRift-Prinzip mehrere Klassen verwendet.

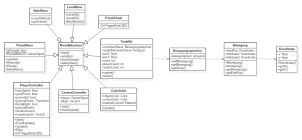


Abbildung 6.1.: UML-Diagramm der Klassen

7. Implementierung

7.1. GUI

Das Griphical User Interface des Spiels wurde sehr einfach gehalten, da es sich um eine Prototyp handelt.

7.1.1. Hauptmenü

Das Hauptmenü besteht aus einem Text und zwei Buttons, zu sehen in Abb.7.1. Damit das Menü das gezeigte Aussehen hat sind folgende Arbeitsschritte zu machen

Abbildung 7.1.: Hauptmenü

Als erstes wird eine leere Szene benötigt. Dafür kann einfach die Tastenkombination Strg. + C gedrückt werden oder unter dem Menü links oben von Unity-Editor File –> New Scene.

Wenn die neue Szenen angelegt ist mittels Rechtsklick in der Hierarchie der Szene eine neue Panel-Komponente anlegen, Abbildung 7.2. Durch das Anlegen eines Panels wird auch automatisch eine Canvas-Komponente erstellt, welches hierarchisch gesehen über dem Panel steht.

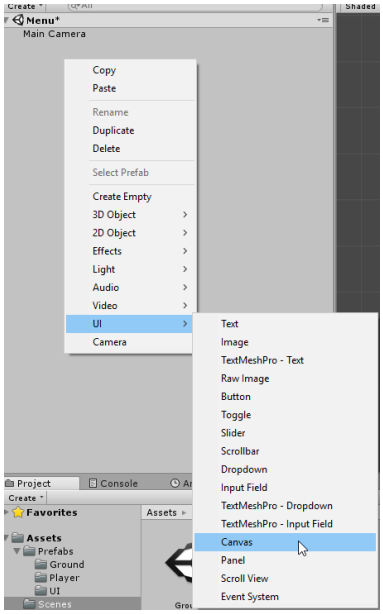


Abbildung 7.2.: Anlegen einer Panel-Komponente

Bei der Canvas-Komponenten müssen die Einstellungen von Abb. 7.3 getroffen werden damit sie das gewünschte verhalten erbringt.

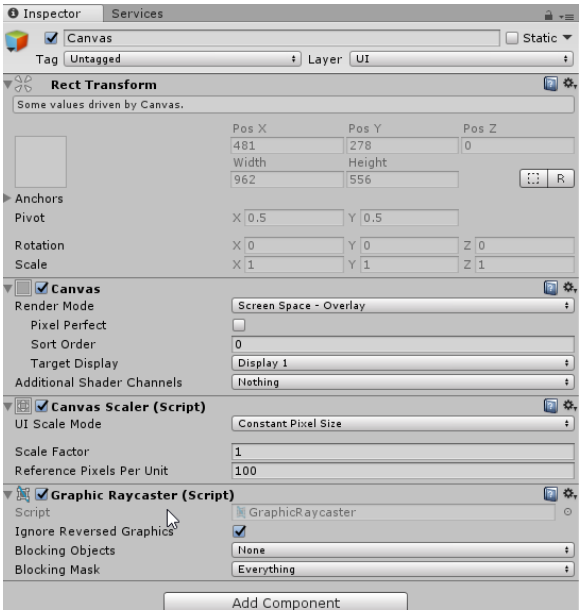


Abbildung 7.3.: Einstellung des Canvas-Komponente im Hauptmenü

Wenn diese Einstellung alle übereinstimmen ist der nächste schritt das bearbeiten des Panels. Hier muss al erstes die Hintergrundfarbe des Menüs festgelegt werden. Dies ist möglich unter bei der Komponente 'Image (Script)'. Wahlweise kann nun eine Grafik als Attribut 'Source Image' gewählt werden oder man verwendet nur eine Farbe wie in unserem Fall um den Hintergrund des Menüs darzustellen. Darüber hinaus wäre es noch möglich einen speziellen Effekt zu beim Hintergrund zu erzielen mittels der 'Material' Eigenschaft. Alle Eigenschaften für die Panel-Komponente sind in Abbildung 7.4 zu sehen.

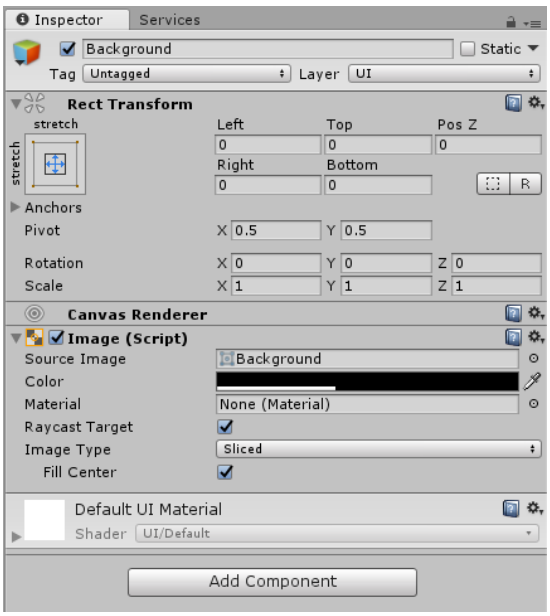


Abbildung 7.4.: Einstellung der Panel-Komponente im Hauptmenü

Sobald die Panel-Komponente fertig konfiguriert ist geht es an die nächste Arbeit. Als erstes wird ein Textfeld erstellt welches mittels Rechtsklick in der Hierarchie links, unter dem UI zu finden ist.

Das Textfeld ist eine einfach aufgebaute Komponente hier wurde ledeglich der Text 'NeXt' beim Attribut Text angegeben und die Schriftgröße auf 30 erhöht, zu sehen in Abbildung 7.5

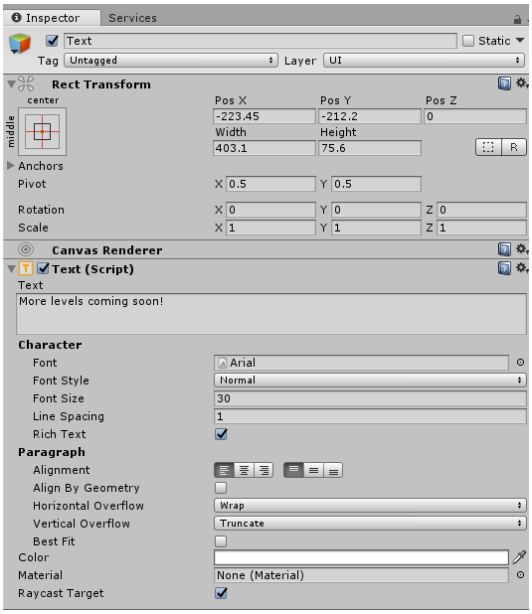


Abbildung 7.5.: Einstellung der Text-Komponente im Hauptmenü

Darauf hin wurden die Buttons,wie die anderen Komponenten, hinzugefügt. Der Button welcher später zum Level-Menü führt wurde bewusst größer gemacht wie der 'Quit'-Button der das Spiel schließt, da so dem Spieler die Auswahl der Levels besser in Auge sticht. Eine Button-Komponente hat in der Hierarchie unter sich immer eine Text-Komponente, wenn sie neu erstellt wird, Diese Textfeld ist die Beschriftung des Buttons. Da wir uns für einen grauen Hintergrund entschieden wurde der Text der Buttons weiß. Die Buttons selbst haben, je nach Position des Mauszeigers, verschiedene Farben . Für das Attribut 'Normal Color' wählten wir schwarz und wenn der Button gedrückt wird ändert sich seine Farbe auf ein dunkles Grau (Attribut: 'Pressed Color'). Dieses verhalten wird in Abbildung 7.6 veranschaulicht.

Abbildung 7.6.: Farbwechsel des Buttons im Hauptmenü beim Klicken

Ist das Einreichen soweit erreicht geht es darum das Verhalten der Buttons bei einem Click-Event zu steuern. Damit für jeden Button nicht extra ein Skript erstellt werden muss erstellen wir als erstes eine sogenannte 'Empty Komponente' welche wie der Name schon sagt leer ist. Danach werden die beiden Buttons in der Hierarchie unter die neu generierte Komponente gestellt. Dieser Schritt ist in Abb. 7.7 zu sehen wobei 'MainMenu' die Empty-Komponente, 'Background' das Panel und 'LevelsButton' sowie 'QuitButton' die Buttons darstellt.

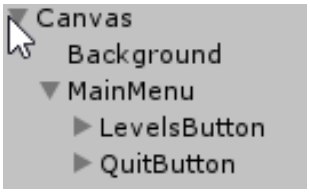


Abbildung 7.7.: Teil der Hierarchie des UI im Hauptmenü.

Nun wurde ein neues Skript im gewünschten Ordner angelegt bei uns hat er den Namen 'Script'. In der Projektstruktur unten einfach mittels Rechtsklick ein C# Skript anlegen (Abbildung 7.8). Sobald es erstellt ist muss es nur noch Programm Code hinein.

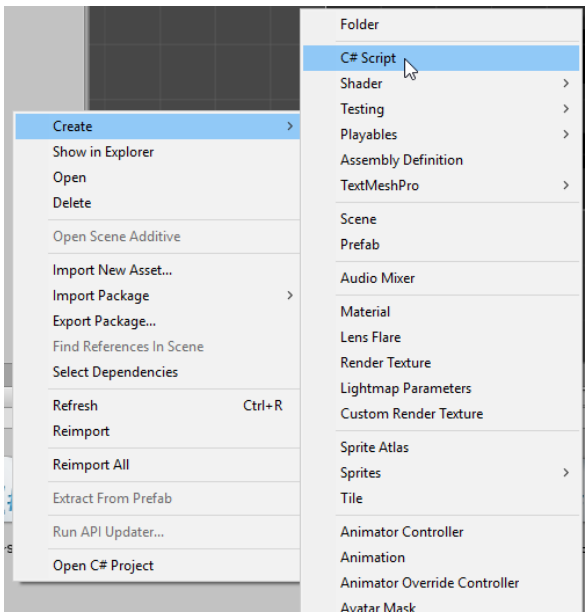


Abbildung 7.8.: Anlegen eines C# Scripts im Unity-Editor

Im Script 'MainMenu' ist der folgende (Listing 7.1) Code Enthalten.

Quelltext 7.1: MainMenu-Script

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class MainMenu : MonoBehaviour
7  {
8
9      public void LevelsMenu()
10     {
11         SceneManager.LoadScene(1);
12     }
13
14     public void quitGame()
15     {
16
17         Application.Quit();
18     }
19 }
```

Die Methode 'LevelsMenu' wechselt die Szene vom Hauptmenü des Spiels zur Szene 'LevelMenu' dies wird über die Methode 'Load Scene(1)' gemacht. Die Zahl 1 die als Parameter mit gegeben wird steht für den Index des Levels Menüs welcher in den 'Build Settings' (Abb. 7.9) festgelegt werden kann. Die Methode 'quitGame()' schließt die gesamte Applikation.

Abbildung 7.9.: Ausschnitt der Build Settings des Unity-Editors

Ist das Skript abgespeichert können nun die einzelnen Methoden den jeweiligen Buttons zugewiesen werden. Dafür im Inspector des Buttons in der Komponente 'Button (Script)' nach 'on Click()' suchen. Als Objekt muss nun das Script 'MainMenu' angegeben und die passende Methode ausgewählt werden, zu sehen in Abbildung **??**. Damit wäre das Hautmenü komplett implementiert.

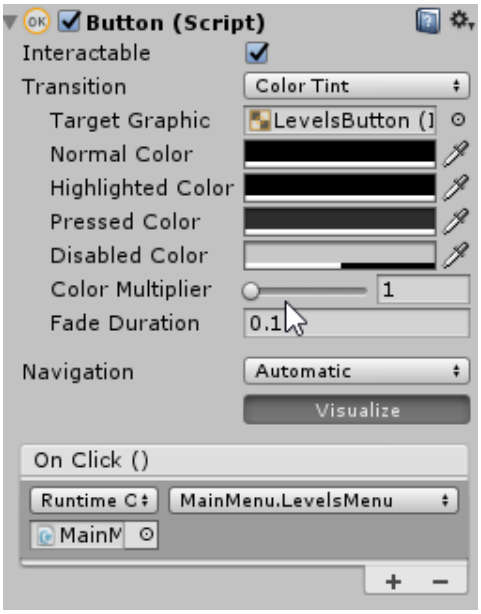


Abbildung 7.10.: Einstellungen der 'Button (Script)' Komponente für den Levels-Button im Hauptmenü

7.2. Leveldesign

7.2.1. Player

Das Level Design ist durch Unity sehr einfach gestaltet. Alle verwenden Graphiken sind nur als Platzhalter verwendet worden.

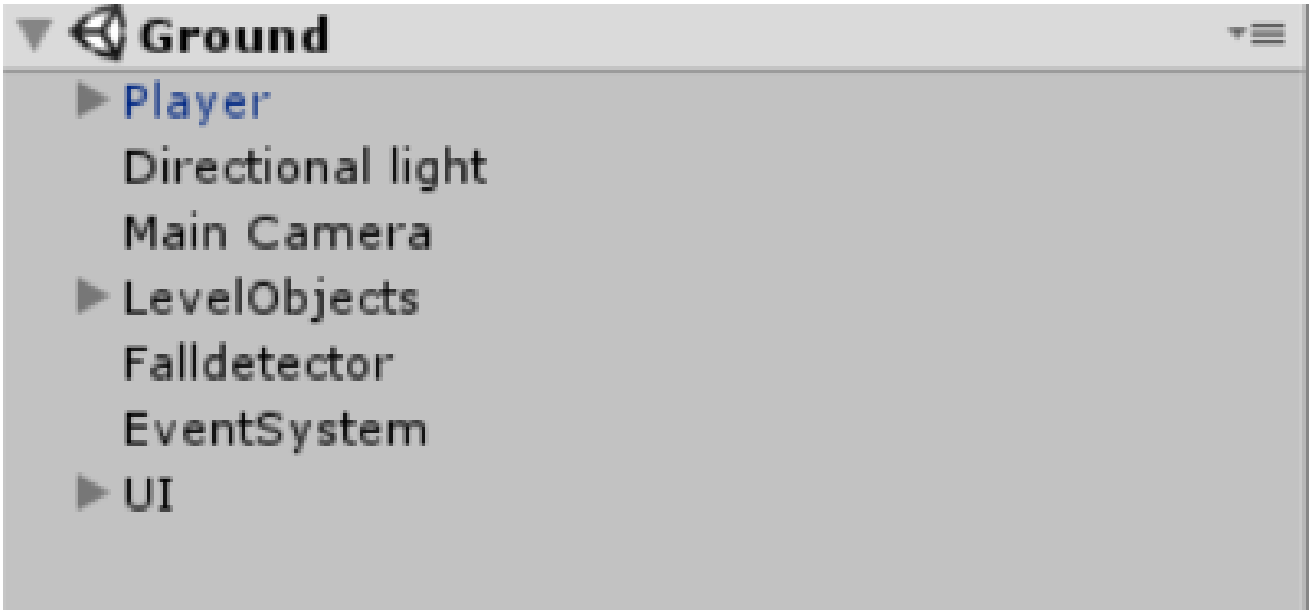


Abbildung 7.11.: Hierarchie der Ground Szene

Für den Player wurde ein Cube-Objekt erstellt.

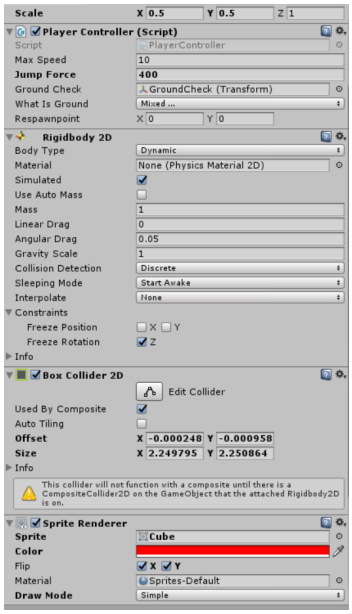


Abbildung 7.12.: Komponenten des Player-Objekt

Der Player besteht aus mehreren Komponenten. Unter anderem der Box Collider 2D, welcher für die Kollisionen zuständig ist. Der Box Collider muss auf die Größe des Players angepasst werden.

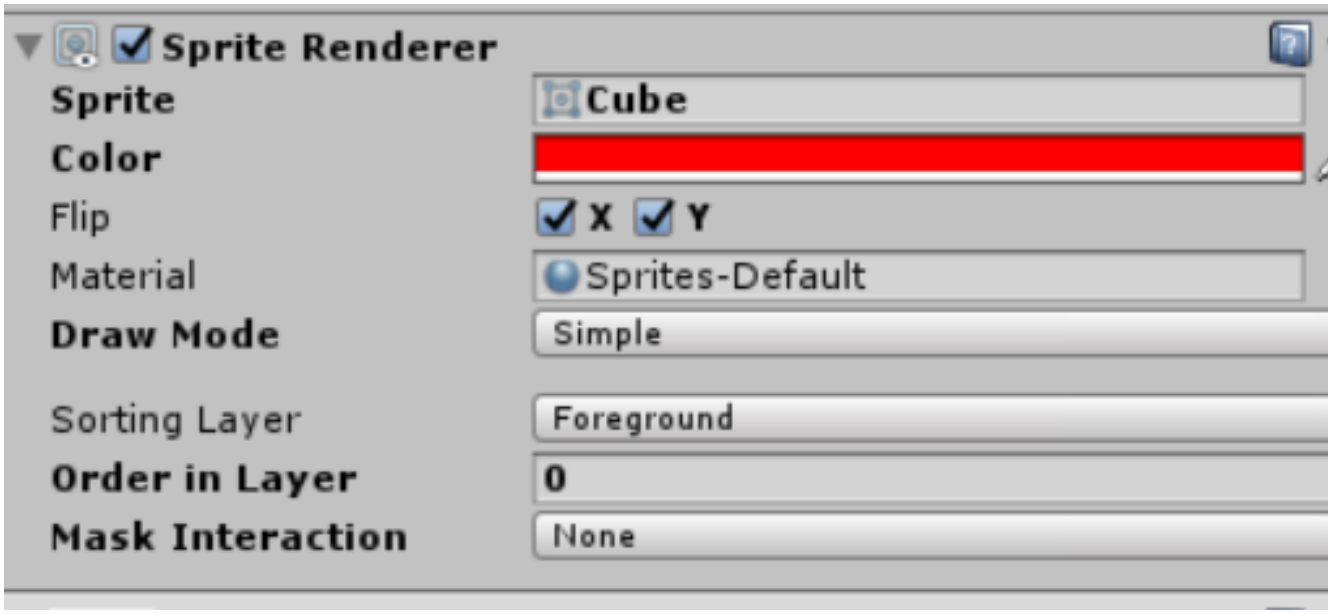


Abbildung 7.13.: Sprite Renderer Komponente von Player

Die Sprite Renderer Komponente ist für die Graphische Darstellung des Players verantwortlich. In diesem Fall ist dies eine durchgehend rote Farbe.

7.2.2. Directional light

Damit man als Spieler überhaupt was sehen kann, muss eine Lichtquelle in dem Level platziert werden. Bild von Komponenten In der Light-Komponente können verschiedene Eigenschaften verändert werden, wie zum Beispiel die Intensity, welche für die Stärke des Lichtes verantwortlich ist.

7.2.3. Main Camera

Ein weiterer Faktor zu sehen des Levels ist die Kamera. Wichtig hier ist, dass man eine geeignete Entfernung wählt, damit man nicht zu wenig oder zu viel sehen kann.

7.2.4. LevelObjects

Bei den LevelObjects handelt es sich um Objekte die Im Level vorkommen, wie der Boden oder Wände. Hierzu werden einzelne Cube-Objekte erschaffen und diese zu Gruppe verbunden. In der Gruppe wird dann der Box Collider 2D hinzugefügt, damit die Objekte eine Kollision mit dem Spieler haben. Auch aber das Ziel zählt unter LevelObjects, wie hier eine Kiste als Platzhalter. Bei der Kiste wurde nur noch ein Skript beigehängt, welches dafür sorgt, dass wenn der Spieler das Ziel erreicht, er zurück zum Level Menü kommt.

7.2.5. Falldetector

Bei dem Falldetector handelt es sich um eine Unsichtbares Objekt, welches ebenso einen Box Collider hat. Das Skript das dem Player angehängt ist, erkennt ob er

dieses Objekt berührt und setzt den Spieler auf die Startposition zurück.

7.3. Unity-Klassenstruktur

7.3.1. MonoBehaviour

Die Klasse „MonoBehaviour“ ist eine Standardklasse, die von der Unity3D-Engine bereitgestellt wird. Jedes Skript, welches in Unity verwendet wird, muss von dieser Klasse erben (Ähnlich wie in Java die „Object-Klasse“ von jeder Klasse gerbt wird). MonoBehaviour gibt folgende Methoden zur Verfügung:

- Start()
- Update()
- FixedUpdate()
- LateUpdate()
- OnGUI()
- OnDisable()
- OnEnable()

Verwendet werden in diesem Projekt aber nur die Start, Update, FixedUpdate und LateUpdate Methoden.

7.3.2. Start-Methode

Die Start-Methode wird nur einmal in der Skriptlaufzeit aufgerufen und zwar bei der Initialisierung des Skriptes.

7.3.3. Update-Methode

Die Update-Methode wird bei jedem Frame aufgerufen. Sie wird meistens für Berechnungen, die in jedem Frame durchgeführt werden müssen, verwendet. Nachteil hier ist aber, wenn das Spiel eine schlechte Framezahl hat werden die Berechnungen auch weniger oft durchgeführt.

7.3.4. FixedUpdate-Methode

Die FixedUpdate-Methode wird bei jedem fixierten Framerate Frame aufgerufen. Diese Methode soll verwendet werden für Bewegungen eines Charakters (Spielbar und nicht Spielbar). Dies hat den Grund, dass selbst bei schlechter Framerate, die Berechnungen der Bewegung konstant bleiben.

7.3.5. LateUpdate-Methode

Die LateUpdate-Methode wird nach allen anderen Update-Methoden aufgerufen. Sie wird hauptsächlich für Funktionen verwendet die nach all den Update-Methoden aufgerufen werden soll z.b.: Für die Kameraführung, da sich Objekte in der Update-Methode bewegt haben können.

7.4. Spieler

Der Spieler steuert mehrere spielbare Charaktere im laufe eines Levels. Damit aber auch die Eingaben des Spieler verwertet werden, werden Skripte benötigt, die diese eingaben auswerten und verwerten. Diese Aufgabe wird von dem PlayerController Skript übernommen.

7.4.1. PlayerController

Damit der Charakter überhaupt Bewegungen ausführen kann, wurde ihm ein Rigidbody2D-Container angehängt. In diesem Container können verschiedene Eigenschaften, die

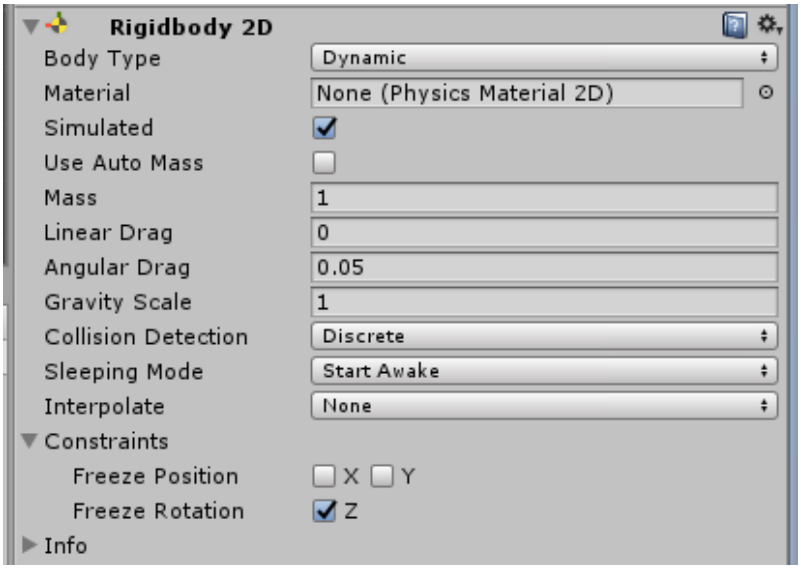


Abbildung 7.14.: Aufbau des Ridigbody2D Containers

sich auf den Charakter auswirken, verändert werden. Unter anderem die "MassËigenschaft, welche für das Gewicht des Charakters zuständig ist.

```
1  void FixedUpdate ()
2  {
3      grounded = Physics2D.OverlapCircle(groundCheck.position,
       groundRadius, whatIsGround);
4
5      float move = Input.GetAxis("Horizontal");
6      Rigidbody2D rb = GetComponent<Rigidbody2D>();
7      rb.velocity = new Vector2(move * maxSpeed,
       GetComponent<Rigidbody2D>().velocity.y);
8
9      if (move > 0 && !faceingRight)
10     Flip();
11     else if (move < 0 && faceingRight)
12     Flip();
13 }
```

Wie schon bei Punkt 7.1.4 besprochen, wurde hier die FixedUpdate-Methode verwendet, da eine Bewegung unabhängig von der Framezahl berechnet werden soll. Am Anfang der Methode wird überprüft ob sich der Charakter auf dem Boden befinden. Hierfür wird der Funktion ÖverlapCircle"die Parameter für den "groundCheck", "groundRadiusünd "whatIsGround"mitgegeben

- Das "groundCheckÖbjekt ist ein unsichtbares Objekt, das dem Charakter angehängt worden und ist dafür zuständig, das sich der Boden und der "groundCheck"überschneiden.
- "groundRadius"legt den Radius fest in dem sich der Boden und "groundCheck"überschneiden müssen, damit die Funktion einen boolschen "true"Wert zurückgibt.
- "whatisGround"hat die Funktion, der Funktion zu sagen, welche Objekte überhaupt als Boden zählen.

Diese Zeile dient dazu, das der Charakter springen kann.

Die Zeilen 6-13 sind für die Bewegung zuständig. In Zeile 6 wird der Input des Spielers eingelesen, wobei nur der Input für die Horizontale Bewegung wahrgenommen wird. In der Zeile 8 wird dann die Bewegung mittels einer Vektor Berechnung ausgeführt. Die Zeilen 10-13 sind für die Änderung der Richtung verantwortlich, wobei beide aber nur die Flip-Methode aufrufen. In der Flip-Methode wird letztendlich nur der Charakter gedreht, damit er sich in eine andere Richtung bewegen kann. Für das Springen des Charakters ist ebenso eine Vektorrechnung verantwortlich, diesmal in der Vertikalen Richtung.

7.5. Kameraführung

Die Kameraführung wurde einfach gelöst. Die Kamera wurde, auf einer bestimmten Entfernung, an den Charakter gehängt.

```
1 public class CameraController : MonoBehaviour
2 {
3
4     public GameObject player;
```

```
5
6       private Vector3 offset;
7
8       void Start ()
9       {
10          offset = transform.position - player.transform.position;
11      }
12
13      void LateUpdate ()
14      {
15          transform.position = player.transform.position + offset;
16      }
17  }
```

Damit die Kamera Flüssig den Charakter verfolgt, wurde die LateUpdate-Methode verwendet.

7.6. Schalter und Türen

Für das Öffnen einer Tür, wurde eine Druckplatte verwendet.

```
1  void Update ()
2  {
3      if ((obj1.transform.position -
       player.transform.position).magnitude < 1.0f)
4      {
5          float step = speed * Time.deltaTime;
6          Door.transform.position =
       Vector3.MoveTowards(Door.transform.position,
       target.position, step);
7      }
8  }
```

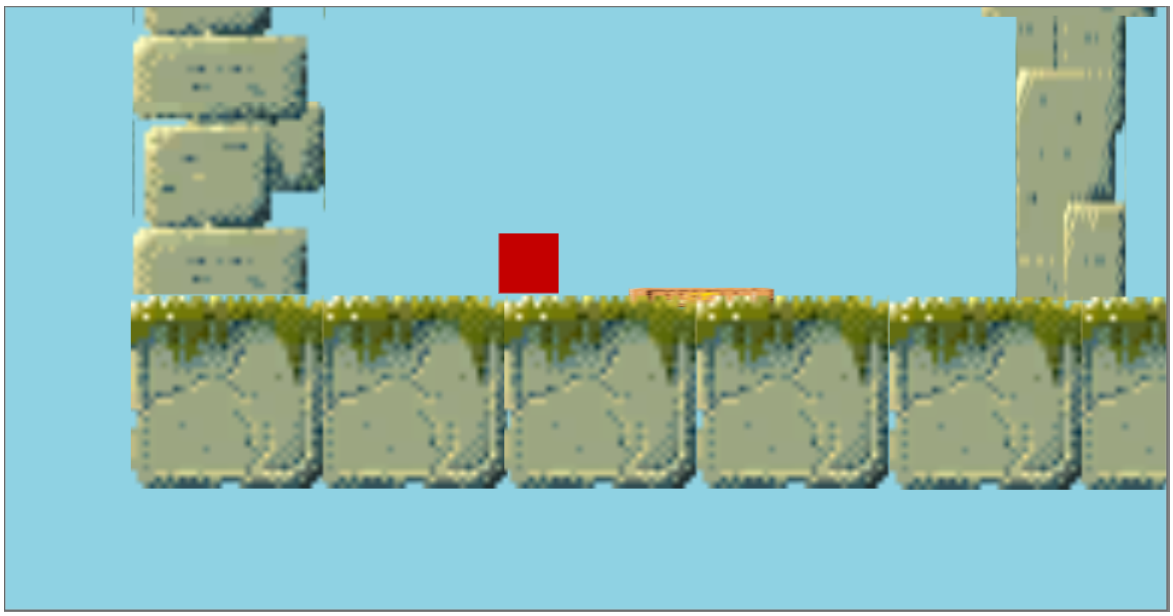


Abbildung 7.15.: Tür geschlossen

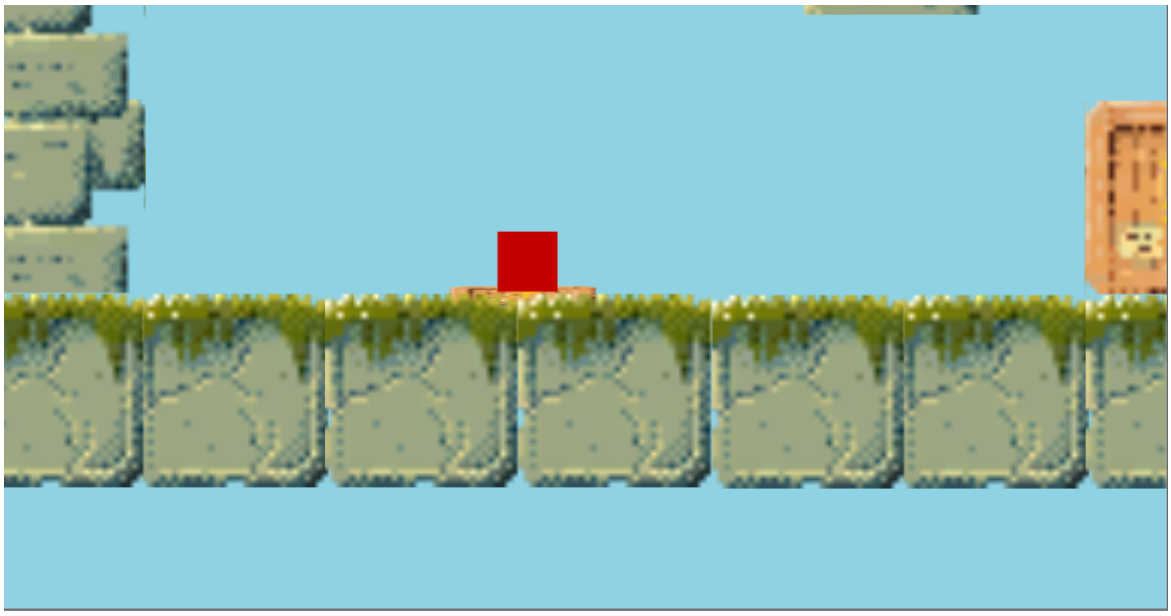


Abbildung 7.16.: Tür geöffnet

7.7. TimeRift

Das TimeRift-Prinzip ist die Grundlegende Funktion in diesem Projekt. Während des erstens Durchlaufes des Spielers, werden in jedem Frame die Koordinaten des Charakters gespeichert.

```
1 void Update ()
2 {
3     movementSave.addBewegung(new Bewegung(new Koordinatee(posX,
      posY), new Koordinate(player.transform.position.x,
      player.transform.position.y)));
4     posX = player.transform.position.x;
5     posY = player.transform.position.y;
6 }
```

Dazu wird eine in der Liste "movementSaveëine neue Bewegung gespeichert. Die Bewegung wird immer von den alten Koordinaten aus gespeichert. Ist jetzt dann der Durchlauf beendet, so wird das Level zurückgesetzt und die Liste "movement-Save"wird und die Liste "restartMovementSave"gespeichert.

```
1 private void restart()
2 {
3     restartMovementSave.Add(movementSave);
4     posX = 0;
5     posY = 0;
6     count = -1;
7     restartCount++;
8     playerCount++;
9     movementSave = new Bewegungsspeicher();
10    movementSave.addBewegung(new Bewegung(new Koordinate(0, 0),
      new Koordinate(0, 0)));
11 }
```

Ebenso werden die andere Werte gesetzt, wie z.B.: Die Startkoordinaten.

```
1 public class Bewegungsspeicher
2 {
3     private ArrayList bewegungslist = new ArrayList();
4
5     public void addBewegung(Bewegung bewegung)
```

```
6       {
7           bewegungslist.Add(bewegung);
8       }
9
10      public Bewegung getBewegung(int zahl)
11      {
12          if ((bewegungslist.Count - 1) >= zahl)
13          {
14              return (Bewegung) bewegungslist[zahl];
15          }
16
17          return null;
18      }
19
20      public ArrayList getArray()
21      {
22          return bewegungslist;
23      }
24  }
```

In der Klasse "Bewegungsspeicher"wird die ArrayList für die Bewegungen Initialisiert.

Damit im Zweiten Durchlauf des Levels, die erste Spielfigur die selben Bewegungen macht, werden in jedem Frame die Koordinaten aus der ArrayList gelesen und bei der Figur gesetzt.

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

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	• Spiel muss gestartet sein • Level muss gewählt sein
Tester	Lukas Vogel
Datum	8.5.2018
Vorgehen	*Soll-Verhalten:* • 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) *Ist Verhalten:* • 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	*Gewünschtes Verhalten:* • Flüssige Bewegung ohne Stopps • Bedingte Beeinflussung des Verhaltens durch die Physik-Engine • Tatsächliche Bewegung durch Tastendruck

Tabelle 8.1.: Testfall 1: Bewegungssteuerung

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 anders Level öffnen will oder mittels Button ins Hauptmenü der Anwendung.
Vorbedingung	• Spiel muss gestartet sein • Level muss gewählt sein • Spieler muss Ende des Levels erreichen
Tester	Lukas Vogel
Datum	8.5.2018
Vorgehen	*Soll-Verhalten:* • 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. *Ist Verhalten:* • Wechsel in die Levelübersicht nach Abschluss des Levels • Wird Levelbutton gedrückt startet richtiges Level • Back-Button öffnet Hauptmenü
Erfolgskriterien	*Gewünschtes Verhalten:* • Der Spieler soll durch das Overlay die Auswahlmöglichkeit haben, weiter zu spielen oder zurück ins Hauptmenü zu navigieren.

Tabelle 8.2.: Testfall 2: Levelabschluss

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	• Spiel muss gestartet sein • Level muss gewählt sein • Spieler fällt aus der Spielwelt
Tester	Michael Leitner
Datum	8.5.2018
Vorgehen	*Soll-Verhalten:* • Charakter wird ab einer bestimmten Grenze („Deathzone“) an den Startpunkt des Levels zurückgesetzt („Respawn“). *Ist Verhalten:* • Respawn der Spielfigur funktioniert.
Erfolgskriterien	*Gewünschtes Verhalten:* • Durch das zurücksetzten des Charakters soll ein reibungsloses weiterspielen möglich sein.

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

Nummer	4
Name	Schalter zum Öffnen einer Tür
Beschreibung	Beim Betätigen des Schalters mittels der Spielfigur, soll eine Tür/Passage geöffnet werden, welche für denn weiteren Spielverlauf wichtig ist.
Vorbedingung	• Spiel muss gestartet sein • Level muss gewählt sein • Spieler ist bei einem Schalter
Tester	Michael Leitner
Datum	8.5.2018
Vorgehen	*Soll-Verhalten:* • Spieler bewegt Spielfigur auf den Schalter • Tür öffnet sich, sobald der Schalter betätigt wurde. *Ist Verhalten:* • Spieler benutzt Schalter und Tür geht auf.
Erfolgskriterien	*Gewünschtes Verhalten:* • Durch das öffnen der Tür sollen neue Gebiete des Levels erschlossen werden, um auch letztendlich das Level abzuschließen zu können

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. 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.

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 Spiele 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 verschieden 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

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.

Literaturverzeichnis

[Schwab 2013] Schwab, Felix: *Systemplanung und Projektentwicklung - Projektentmanagement 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`. – [Online; Stand 23. Juni 2018]

A. Anhang-Kapitel

A.1. Anhang-Section

Testtext