Destiny’s Reminder – Eine interaktive Geschichte



**Ein Projekt von:** Carole Meier

**Abteilung:** G4C

**Datum:** 31.8.23

**Schule:** Kantonsschule Wettingen (KSWE)

**Betreuende Lehrperson:** Oliver Probst

**Gegenleserin:** Sara Nyffenegger   
Zusammenfassung  
In dieser Arbeit wurde die Form der Videospielart Visual-Novel genauer untersucht und ein eigenes Spiel dieser Art hergestellt. Das Ziel war das Schreiben und Gestalten einer interessanten, interaktiven Geschichte, welche dann durch das Programmieren eines Visual-Novels spielbar gemacht wurde. Um herauszufinden, aus welchen Bestandteilen ein solches Spiel aufgebaut ist und wie man eine ansprechende interaktive Geschichte mit verschiedenen Enden schreibt, wurde zu Beginn online Recherche zu diesen Themen betrieben. Für die Kreation des Spiels wurde sich mit der Programmier-Engine Ren‘ Py auseinandergesetzt. Mit den erhaltenen Erkenntnissen konnte so in ein vollständiges und spielbares Skript programmiert werden, zu welchem später die passenden visuellen Elemente angefertigt und angefügt wurden.

Vorwort  
Ein eigenes Spiel wie dieses zu schreiben und zu gestalten war schon lange eine meiner Ambitionen. Seit meiner Kindheit befasse ich mich in meiner Freizeit gerne mit Videospielen und Geschichten jeglicher Art. Mich in Erzählungen zu vertiefen aber auch eigene zu kreieren war schon immer eine meiner Lieblingsbeschäftigungen. Neben dem Lesen und Spielen sind mir nämlich auch das Malen und Schreiben sehr wichtig, um mich kreativ auszudrücken. Gemalt habe ich immer schon gerne, doch eigene Geschichten zu entwickeln und zu illustrieren begann ich erst wirklich in den letzten Jahren. Während meiner Zeit and der Kantonsschule konnte ich dank dem breiten Freikursangebot bereits Erfahrungen im Bereich des Literarischen Schreibens sammeln. Und so meine Leidenschaft Geschichten zu erzählen weiter ausleben. Doch für mich wurde während dieser Zeit auch klar, dass ich meine Geschichten gerne auch visuell darstellen möchte. Durch mein Interesse an Informatik und Gamedesign entstand so schlussendlich die Idee für diese Arbeit. Ich wollte eine eigene interaktive Geschichte in Form eines Spiels programmieren.

Inhaltsverzeichnis

[Inhaltsverzeichnis 3](#_Toc147740800)

[1. Einleitung 1](#_Toc147740801)

[1.1 Themeneingrenzung 1](#_Toc147740802)

[1.2 Ziele 1](#_Toc147740803)

[1.3 Fragestellungen 1](#_Toc147740804)

[1.4 Methoden 1](#_Toc147740805)

[2. Theorie/Recherche 1](#_Toc147740806)

[2.1 GitHub 1](#_Toc147740807)

[2.2 Interaktive Geschichten und Visual-Novels 2](#_Toc147740808)

[2.3 Ren.py 2](#_Toc147740809)

[2.4 Welche Bild Elemente sind benötigt? 3](#_Toc147740810)

[3. Arbeitsprozess 5](#_Toc147740811)

[3.1 Konzipierung der Geschichte 5](#_Toc147740812)

[3.2 Planung der Handlungsstränge 6](#_Toc147740813)

[3.3 Schreibprozess des Skripts in Ren.py 6](#_Toc147740814)

[3.3.1 Definieren der Charaktere 6](#_Toc147740815)

[3.3.2 Schreiben der Szenen mittels Labels 7](#_Toc147740816)

[3.3.3 Einfügen Interaktiver Elemente 8](#_Toc147740817)

[3.4 Illustrationsprozess der Spielgrafiken 9](#_Toc147740818)

[3.4.1 Gestaltung der Charaktere und diverser Spielelemente 9](#_Toc147740819)

[3.4.2 Gestaltung der Hintergründe 10](#_Toc147740820)

[3.4.3 Gestaltung der Benutzeroberfläche und Icons 11](#_Toc147740821)

[3.5 Verknüpfen von Skript und Grafik 12](#_Toc147740822)

[3.5.1 Einfügen eigener Grafiken und Schriftarten 12](#_Toc147740823)

[3.5.2 Darstellen der Charakter und weiterer Spielelemente 12](#_Toc147740824)

[3.5.3 Aufrufen der Hintergründe 12](#_Toc147740825)

[3.5.4 Effekte und Übergänge 12](#_Toc147740826)

[3.5.5 Der Feinschliff 12](#_Toc147740827)

[4.Schluss 13](#_Toc147740828)

[4.1 Beantwortung der Fragestellung 13](#_Toc147740829)

[4.1.1 Wie schreibt man eine Interaktive Geschichte? 13](#_Toc147740830)

[4.1.2 Wie setzt man eine Geschichte in eine Spielbare form um/Welche Methoden gibt es? 13](#_Toc147740831)

[4.1.3 Was braucht es, um einen Visual Novell zu kreieren? 13](#_Toc147740832)

[4.1.4 Wie verstärken visuelle Elemente den Eindruck einer Geschichte? 13](#_Toc147740833)

[4.1.5 Wie kann ich meinen Code strukturiert notieren (Clean Code)? 13](#_Toc147740834)

[5. Reflexion 14](#_Toc147740835)

[5.1 Inhalt 14](#_Toc147740836)

[5.2 Methoden 14](#_Toc147740837)

[6. Quellenverzeichnis 14](#_Toc147740838)

[7. Anhang 16](#_Toc147740839)

# 1. Einleitung

## 1.1 Themeneingrenzung

Bla

## Ziele

Das Ziel dieser Arbeit war es, eine ansprechende Interaktive Geschichte zu gestalten und diese durch das Programmieren eines Videospiels in spielbarer Form umzusetzen. Die Geschichte sollte mehrere interessante Pfade und Enden besitzen, welche der Spieler/die Spielerin mit seinen/ihren Entscheidungen beeinflussen kann. Die Geschichte sollte hauptsächlich aus natürlich- und spannend gestaltetem Dialog der verschiedenen Charaktere bestehen und sollte durch passende Grafiken zu einem in sich abschließenden und visuell ansprechenden Spielerlebnis werden.

## 1.3 Fragestellungen

Eine interessante interaktive Geschichte in Form eines Videospiels erstellen zu können erforderte eine ausführliche Auseinandersetzung mit folgenden Fragen:

* Wie schreibt man eine Interaktive Geschichte?
* Wie setzt man eine Geschichte in eine Spielbare form um/Welche Methoden gibt es?
* Was braucht es, um einen Visual Novell zu kreieren?
* Wie verstärken visuelle Elemente den Eindruck einer Geschichte?
* Wie kann ich meinen Code strukturiert notieren (Clean Code)?

## 1.4 Methoden

Da sich die Umsetzung einer Geschichte in spielbarer Form aus einer Kombination vieler verschiedenen Bereichen zusammen setzt waren Recherche und Arbeit in den Bereichen Literatur, Kunst und Informatik nötig. Zur Recherche diente hauptsächlich das Internet. Bezogen wurden verschiedene Dokumentationen der genutzten Programme wie auch Anleitungsvideos zu spezifischen Problemen. Literarische Recherche war nur minimal nötig da bereits eine gute Grundlage und einiges an Erfahrung bezüglich literarischer Arbeit bestand. Als Vorbereitung wurden verschiedene Pläne und Mindmaps der möglichen Geschichte gezeichnet und schlussendlich auf ein Übersichtliches Flussdiagramm der Handlungen der Geschichte ausgearbeitet. Geschrieben wurde die Geschrieben anschließend direkt mit der Visual-Novel-Engine Ren’ Py. Letztlich wurden die nötigen Grafiken direkt Digital mit auf einem iPad in Procreate gemalt und genutzt. Als Hilfsmittel zur Planung und Illustration der Hintergründe diente Sims 4. Der Fortschritt der Arbeit wurde mittels GitHub festgehalten.

# 2. Theorie/Recherche

## 2.1 GitHub

Schon von Beginn an war klar, dass in diesem Projekt selbst geschriebener Code involviert werden würde. Als Werkzeug zum Speichern und Teilen dieses Codes wurde deshalb von der Betreuenden Lehrperson empfohlen GitHub1 zu verwenden. «GitHub ist ein Onlinedienst zur Versionsverwaltung für Software-Entwicklungsprojekte»2. GitHub ist ein weitverbreitetes, kollaboratives Versionsverwaltung Tool, welches heutzutage von vielen Softwareentwicklern und Informatikern verwendet wird, um Code über das Internet zu speichern und mit anderen zu Teilen.

GitHub ist ein nützliches Hilfsmittel für jede Art Projekt, da es ermöglicht kontinuierlich neue Versionen von Codefiles und ganzen Projekten festzuhalten und archivartig abzuspeichern.

Mit einem GitHub Account kann reibungslos von verschiedenen Geräten und Personen and einem Projekt gearbeitet werden, dank ihres Cloud-Storage Systems.

Um die Änderungen eines Files oder eines Ordners aufzeichnen zu können, wird auf GitHub ein Repository generiert. Dieser agiert als Speichermedium aller Dateien, welche von GitHub für dieses spezifische Projekt erfasst werden sollen. Das Programm merkt, sobald eine Änderung innerhalb dieses Ordners vorgenommen wird und zeichnet diese auf. Werden diese Änderungen dann abgeschlossen, kann durch einen Commit-Befehl diese neue Version des Projekts wieder mit der Cloud synchronisiert werden. Die alte Version wird im Archiv abgespeichert und durch die Neue ersetzt. GitHub ist ein Nützliches Werkzeug für jegliche Arbeiten, welche mit Code zu tun haben, da es jede Änderung in seinen Speicher aufnimmt und chronologisch festhält. Falls ein Fehler entsteht oder Code aus älteren Versionen benötigt wird, können durch GitHub diese alten Versionen wiederhergestellt werden. Mit jedem Commit wird das Datum und die genaue Uhrzeit aufgezeichnet und der Nutzer/die Nutzerin wird aufgefordert eine kurze Beschreibung der Änderungen anzufügen. Dies erleichtert es später bestimmte Versionen des veränderten Codes wiederzufinden.

Für dieses Projekt bedeutete die Nutzung von GitHub, dass kein zusätzliches Journal geführt werden musste. Den Code und die weiteren Spieldateien fortlaufend in die Cloud hochzuladen, würde ein traditionelles Journal ersetzen, da jede Änderung bereits auf GitHub aufgezeichnet wird. Durch das Teilen des Projekt Repositorys mit der Betreuenden Lehrperson konnte so auch der Fortschritt der Arbeit dokumentiert werden.

## 2.2 Interaktive Geschichten und Visual-Novels

Interaktive Geschichten existieren in vielen Formen und für jederlei Altersgruppen. «Choose your own Adventure» Bücher wie zum Beispiel die 1000 Gefahren Serie des Ravensburger Verlags3, sind zwar in Stil und Inhalt eher an Kinder gerichtet, zeigen jedoch erfolgreich wie auch in rein Text basierenden Medien ein spielerisches Leseerlebnis entstehen kann. Eine weitere Methode findet man auch in digitale Medien, wie Film und Videogames. «Bandersnatch»4 aus der Serie «Black Mirror» setzt die Idee einer selbst gewählten Geschichte erfolgreich in einem Film um. Inspiration für dieses Projekt wurde jedoch aus dem letzteren Bereich der digitalen Medien gezogen. Als Ziel der Arbeit wurde ein kurzer selbst illustrierter Visual-Novel gesetzt.

Visual-Novel:

Wie der Begriff bereits impliziert, besteht ein Visual-Novel aus einer Kombination von Text und Grafik, welche zusammen eine spielbare Geschichte bilden. Oft wird das Genre als Unterklasse der Videospiele gesehen, da sie ebenfalls eine Art digitaler, interaktiver Erzählungen sind. Seinen Ursprung nahm diese Spielart in Japan, wo sie auch heute noch sehr beliebt sind. Obwohl sie mittlerweile auch in anderen Ländern bekannt sind, ist der Einfluss Japanischer Kultur wie auch der von Manga und Anime oft noch gut erkennbar. Beispiele bekannter Visual-Novels wären die «Persona»5 Serie von ATLUS und SEGA oder «Doki Doki Literature Club»6 welches von einem kleineren, Indie-Studio namens Salvato programmiert wurde. Dank des simplen Aufbaus eignet sich ein Visual-Novel als Projekt für kleinere Studios oder Einzelpersonen. Ebenfalls eignen sie sich gut als Einstiegprojekt um sich mit dem Designen eines Spiels bekannt zu machen, da sie den Fokus auf eine zentrale Erzählung setzten und diese hauptsächlich durch sehr limitiertes dialogbasiertes Gameplay umsetzten. Im Gegensatz zu anderen Videospielen interagiert der Spieler/die Spielerin in einem Visual-Novel über eine direkte Ich-Perspektive mit der Geschichte. Meist wird mit der Hilfe von Klicks durch die Gespräche navigiert und teils auch durch eigene Entscheidungen beeinflusst. Auf Grund des sehr einseitigen Gameplay, welches primär aus Lesen besteht, werden Visual-Novels manchmal gänzlich von Videospielen getrennt.7 Sie klar zu einem der vielen erzählerischen Medien zu zählen ist jedoch schwierig, da Visual-Novels viele Aspekte verschiedener solcher Medien mischen und nutzen um eine neue Form der Erzählung zu generieren. Mir ihren limitierten, vorgefertigten Hintergründen und Charakteren und der dialogbasierten Erzählungsweise sind Visual-Novels Dramen, in ihrer vorgeführten Form, sehr ähnlich. Im Gegensatz zu anderen literarischen Erzählungsformen beinhalten sie keine beschreibende Textstellen oder Begleitsätze. Stattdessen verlassen sie sich auf Visuelle Elemente um Gefühle und Atmosphären zu vermitteln. Was in einem Drama zum beispiel jedoch gesprochen wird, wird in einem Visual-Novel vom Spieler/der Spielerin in Form von Dialog-Text-Boxen gelesen. Sie mischen so!!!!!

## 2.3 Ren.py

Früh in der Eingrenzungsphase dieser Arbeit, fiel die Entscheidung die Visual-Novel Engine Ren’ Py8 zu nutzen, um schlussendlich das Spiel zu kreieren. Ren’ Py ist eine auf Python basierende Programmier-Engine, welche eine gute Grundlage bietet, um eigene Interaktive Geschichte in Form eines Videospiels zu erstellen. Über 4'000 spiele wurden bereits mit der Engine erstellt und sind zurzeit im Internet erhältlich.9 Spiele wie «Doki Doki Literature Club», welches bereits zuvor erwähnt wurde, entstanden ebenfalls mit der Hilfe von Ren’ Py. Weshalb sie auch als Inspiration für mögliche Themen und Spielmechaniken, in Destiny’s Reminder dienten. Zusätzlich wurde schnell ersichtlich, dass die Kombination des Python-codes mit den bereits vordefinierten Befehlen, welche Ren’ Py seinen Nutzern/Nutzerinnen bietet, einen verständlichen und äusserst intuitiven Schreibprozess ermöglichen würde und eignete sich somit perfekt für dieses Projekt.

Ren’ Py stellt für jedes Projekt eine Vorlage zur Verfügung welche zu belieben verändert und auf das geplante Spiel angepasst werden kann. Es erlaubt eigene Bilder einzufügen und macht es einfach diese mit dem Code zu verknüpfen. Dafür erstellt das Programm für jedes Projekt einen Ordner, in welchen jegliche Spieldateien und Bilder abgelegt werden, um später vom Code genutzt zu werden. Dieser Ordner konnte sehr einfach zur Aufzeichnung des Fortschrittes und für Backups in einem GitHub Repository platziert werden, da dieser alle nötigen Dateien enthält, um später das Spiel starten zu können. Der Code, welcher schlussendlich das Spiel bildet, wird von Ren’ Py in vier verschiedene rpy-Dateien unterteilt, welche jeweils einen der Aspekte des Gamedesign Prozesses enthalten. Ren’ Py lässt seine Nutzer/Nutzerinnen die komplette Freiheit, wenn es um die Gestaltung der einzelnen Komponenten eines Spiels geht und kann sogar genutzt werden, um am Ende eine App-Version des eigenen Projektes zu generieren.

## 2.4 Welche Bild Elemente sind benötigt?

Wie in Abschnitt 2.2 erwähnt, sind die Bilder und Animationen ein essenzieller Teil eines Visual Novels. Das Darstellen von Bildelementen in Ren’ Py ist grundsätzlich keineswegs limitiert. Es können beliebig viele Grafiken und Animationen dargestellt werden und sogar die Benutzeroberfläche kann selbst angepasst werden.Welche Form die Visuellen Elemente für dieses spezifische Projekt nehmen mussten klärte sich erst nach einem genaueren Blick auf die Funktionen und Prozesse der Ren’ Py-Engine. Um eine genauere Idee zu erhalten welche Illustrationen für einen simplen Visual-Novel wie diesen benötigt werden wurden die Darstellungsweisen der Grafiken in 4 Klassen unterteilt.

Einfache Objekte:

Dies ist die einfachste der 4 Darstellungsweisen und wird benutzt, um jegliche nicht interagier bare Objekte darzustellen. Um eine bestimmte Grafik so darzustellen, muss im Code lediglich ihr Name aufgerufen werden. Das bedeutet, dass ein beliebiges Bild, gezeigt werden kann. Man findet hier keine Limitationen bezüglich Form und Grösse. Charaktere wie auch andere einfache Objekte können so mit einer Vielzahl verschiedener darstellungs-Einstellungen, wie zum Beispiel Position, Grösse oder Transparent zu einem Beliebigen Zeitpunkt durch den show-Befehl aufgerufen und durch einen Hide-Befehl wieder versteckt werden. Diese Art des Darstellens limitiert nicht die Anzahl der sichtbaren Objekte. Und diese können in beliebiger Reihenfolge hintereinanderstehen.

Benutzeroberfläche:

Diese Art von Grafik ist sehr ähnlich wie die der einfachen Objekte. Im Grunde unterscheiden sich die beiden gar nicht. Um das Aussehen von Elementen wie TextBoxen, Menus oder Icons anzupassen müssen nur die jeweiligen Grafiken im Projektordner ausgetauscht werden. Man kann so komplett selbst entscheiden wie zum Beispiel der Dialog im Spiel dargestellt wird und welche Form, Grösse etc. die TextBoxen und Menus haben. Diese Darstellungsweise wurden jedoch für dieses Projekt in eine separate Klasse eingeteilt, da sie nicht selbst im Code aufgerufen werden müssen und lediglich durch das ersetzten von vordefinierten Objekten erfolgten. Diese Bilder werden nicht durch das Skript, sondern durch einen separaten Teil des Codes, welcher unter GUI gespeichert ist, definiert und wurden um für dieses Spiel nur minimal angepasst, da der Fokus dieser Arbeit auf der Geschichte und dem Umsetzen des eigentlichen Spielerlebnis lag.

Hintergründe:

Wie auch die Elemente der Benutzeroberfläche, ähneln die Hintergründe den einfachen Objekten in vielen Aspekten. Sie werden gleich wie die Einfachen Objekte direkt durch das Skript aufgerufen, haben jedoch ein paar spezielle Eigenschaften. Dank der Weise, in welcher sie aufgerufen werden, beeinflussen sie die anderen Grafiken, welche zur Zeit vom Spiel dargestellt werden. Wenn ein Hintergrund durch den Befehl Scene aufgerufen wird, sorgt dies dafür, dass jegliche zuvor dargestellte Visuellen Objekte verschwinden. So kann durch einen Scene-Befehl sowohl ein neuer Hintergrund gesetzt werden wie auch direkt eine neue, leere Szene begonnen werden. Hintergründe haben ebenfalls eine Format Anforderung. Um korrekt vom Spiel dargestellt werden zu können müssen die Hintergrund Illustrationen bereits in derselben Grösse und Auflösung wie das Spiel gegeben werden.

Animationen und Videosequenzen:

Die letzte und speziellste Gruppe bilden die Animationen und Videosequenzen. Ren’ Py kann nämlich nicht nur Bilder, sondern auch ganze Videodateien darstellen. Dies geschieht jedoch über einen separaten komplizierteren Befehl, welcher in diesem Projekt nicht angewendet wurde.

# 3. Arbeitsprozess

## 3.1 Konzipierung der Geschichte

Handlung:

Bevor mit dem eigentlichen Schreibprozess begonnen werden konnte, war eine packende Idee nötig, die die Handlungen des Spiels später tragen würde. Diese Idee kam in Form einer bereits begonnenen Geschichte, welche das Kern-Konzept der Täglichen Achievements einführte. Während dem Verlauf des Spiels würde der Hauptcharakter Destiny auf einem für andere unsichtbaren Textfeld erfahren, was am kommenden Tag geschehen würde. Die Geschichte verfolgt über 5 Tagen wie der Hauptcharakter lernt wie mit diesen Prophezeiungen umzugehen ist und erfährt, was sie verursacht. Die klare 5 Tage Struktur eignete sich gut für ein Spiel dieser Art, da die Kapitel artigen Etappen dem Spieler/der Spielerin genügend eine klare Struktur mit Abwechslung bieten und gleichzeitig auch genutzt werden konnten, um einen klassischen Spannungsbogen aufzubauen. Die Tage wurden, wie in einem klassischen 5 Akt Drama10 unterteilt, um jeweils die Exposition, die Steigende Handlung, den Höhepunkt, die Fallende Handlung und die Lösung des Konflikts einem der Tage zuteilen zu können.

Um den Umfang des Projekts etwas zu limitieren wurde der Fokus der Geschichte auf 5 Hauptcharaktere gesetzt, von denen eine die Spielbare Figur und somit den Spieler/die Spielerin repräsentiert. Im Laufe der Geschichte erfährt die Hauptfigur Destiny von einer Entführung einer der anderen Charaktere, Kai, und versucht dieser Entführung auf den Grund zu gehen. Im Verlaufe des Spiels soll sich dann der Spieler/die Spielerin, basierend auf Hinweisen in den 5 Tagen, entscheiden, welche der restlichen 3 Figuren in der Entführung involviert waren und wie Destiny der Entführten Person helfen kann. Das Ziel soll sein die entführte Person zu retten und den Täter zu schnappen. Die Geschichte sollte zu Beginn sehr fröhlich und nichts ahnend sein doch sich dann weiter und weiter verwickeln bis zum Teil sogar eine düstere Atmosphäre entsteht.

Charaktere:

Da ein erheblicher Teil der Geschichte auf den Gesprächen zwischen Destiny und diesen 3 Charakteren beruht, wurde großen Wert darauf gelegt jeden einzelnen einzigartig zu Gestalten und ihnen eine individuelle Persönlichkeit zu verleihen. Hierbei wurden die Persönlichkeit der entführten Figur, Kai, und die von Destiny, dem Hauptcharakter, absichtlich eher neutral gehalten. Man erhält so etwas mehr Freiraum sich selbst in diese Figuren hinein zu versetzten. Eine neutralere und somit formbarere Persönlichkeit bedeutete, dass sich die Entscheidungen später natürlicher anfühlen würden, sowohl für den Spieler/die Spielerin wie aber auch für Destiny als Figur. Um jedoch eine solide Grundlage zu legen auf welche die Geschichte aufbauen konnte, wurden dennoch einige ihrer Charakterzüge und Eigenarten eingefügt. Diese würden ihre Interaktionen mit den anderen Figuren zu leiten.

Destiny ist eine eher schüchterne, aber talentierte junge Frau. Manchmal etwas unordentlich und unorganisiert. Diese eher unvorteilhaften Charaktereigenschaften zu zeigen, diente dazu sie dem Spieler/der Spielerin sympathischer zu machen und gleichzeitig ihre Unbeholfenheit in einer neuen Situation zu Illustrieren. Viele der Entscheidungen würden auf kleine Fehler aufbaien welche Destiny in ihrem Alltag geschehen. Während dem Verlauf der Geschichte erfährt man, dass Destiny vor etwa einem Jahr einen neuen Lebensabschnitt bei einer neuen Firma und einem neuen Wohnort begonnen hatte.

Gabriel, Matthew und Simon, die 3 Figuren, mit welchen sich Destiny in den 5 Tagen hauptsächlich unterhält, wurden jeweils gemäß einem der vielen Klischees eines potenziellen Antagonisten geschrieben. Da der Spieler/die Spielerin am Ende der Geschichte entscheiden würde wer der wahre Antagonist war mussten alle mögliche suspekte bleiben. Der Trope des „Falschen besten Freunds“, des „Tyrann“, und des „Wolf im Schafspelz“ können alle erfolgreich genutzt werden, um sowohl den wahren Antagonisten zu verkörpern oder als Falsche Fährte zu dienen. Sie auf diese sehr oberflächlichen und bekannten Klischees zu reduzieren, machte es einfacher starke und klar motivierte Persönlichkeiten zu kreieren. Außerdem helfen diese Stereotypen ebenfalls dem Spieler/der Spielerin bereits früh die verschiedenen Charaktere zu verstehen und ihre weiteren Handlungen abzuschätzen.

Gabriel ist der „Beste Freund“ oder „Falsche beste Freund“. Er ist sehr fürsorglich und immer nett zu Destiny. Er versucht oft das Beste aus einer Situation zu machen und rettet sie aus unangenehmen Situationen. Er und Destiny sind bereits befreundend zum Zeitpunkt zu welchem die Geschichte beginnt und verbringt so natürlich am meisten Zeit mit ihr. Je nach Route, welche der Spieler/die Spielerin einschlägt, hilft Gabriel den Täter zu schnappen. Jedoch kann er auch verdächtigt und von Destiny genauer verfolgt werden.

Matthew füllt die Rolle des „Tyrann“. Als Destiny’s Chef und Leiter der Abteilung passt seine eher sehr kalte und egoistische Art zu seiner Position. Er nutzt oft seine Autorität, um andere einzuschüchtern und dadurch eigene Vorteile zu erzielen. Er ist ernster als die beiden anderen und zeigt offen, dass er Destiny nicht leiden kann. Seine Nazistische Art und einige Auseinandersetzungen zur Entführung können Verdacht erwecken. Oberflächlich passt er am besten zum Stereotypischen Bild eines Bösewichts, hat letztendlich jedoch er nichts mit der Entführung zu tun.

Simon ist der wahre Täter und stellt einen „Wolf im Schafspelz“ dar. Er ist zu Beginn sehr freundlich zu Destiny und hilft ihr sich in ein neues Projekt zu vertiefen. Er scheint stets unbesorgt und bleibt hilfsbereit. Was sich jedoch später als Fassade herausstellt. Er manipuliert viele der Situationen, um harmlos und unwissend zu wirken, zieht im Hintergrund jedoch die Fäden.

## 3.2 Planung der Handlungsstränge

Als die Charaktere und die grobe Handlung des Spiels entschieden waren, galt es die genaueren Details und Verzweigungen der Geschichte festzulegen. Dabei war zu beachten, dass, je stärker eine Entscheidung auf die Handlung wirkte würde oder je häufiger man solche treffen könnte desto persönlicher und somit spannender würde schlussendlich das Spielerlebnis auch sein. Gleichzeitig aber jedoch bei jeder neuen Verzweigung erneuter Schreibaufwand und womöglich neuen Bedarf an Illustrationen entstehen würde. Folglich wurde Wert darauf gelegt die Anzahl und die Konsequenzen der Entscheidungen in einem umsetzbaren Rahmen zu halten und diese gut zu planen, um später Fehler zu vermeiden. Mit einem genauen Plan konnte, während dem Schreiben des Skript-codes der Überblick behalten werden.

Um die Handlung zu planen, eigneten sich erst Mindmaps und später Flussdiagramme, welche die verschiedenen Entscheidungen und die darauffolgenden Pfade klar visuell darstellen konnten. Ein solches wurde auf der „Draw.io“11 generiert. Abbildung 1 zeigt dieses detaillierte Flussdiagramm. Es enthält alle größeren und auch einige der nicht so relevanten Entscheidungen. Der Spieler/Die Spielerin kann zu (!) Zeitpunkten, während dem Verlauf des Spiels zwischen zwei oder mehr Optionen wählen und so den weiteren Verlauf der Geschichte beeinflussen. Wie zuvor genannt, gibt es auch mehrere kleinere Entscheidungen, welche keinen oder minimen Einfluss auf den allgemeinen Verlauf der Geschichte nehmen. Diese sind zwar eher insignifikant, ermöglichen jedoch ein individuelleres und interaktiveres Spielerlebnis, ohne den Zeitaufwand relevant zu erhöhen.

Im Ganzen besitzt die Geschichte (!) verschiedene Wege das Spiel zu beenden und vier haupt Endsituationen, welche so erreicht werden können.

Je nach Entscheidung, welche gefällt wurde, trifft ein sehr schlechtes, schlechtes, gutes oder sehr gutes Ende ein.

Das Ziel des Spielers/der Spielerin sollte sein zu einem möglichst guten Ende zu gelangen. Diese Eingrenzung auf 4 Haupt Situationen half, eine fokussiertere Geschichte zu erzählen und minimierte wiederum den nötigen Schreibaufwand. Die 4 Situationen unterscheiden sich in 2 Aspekten voneinander. Einerseits beeinflussen die Entscheidungen, ob die Entführte Person gefunden wird, andererseits bestimmen sie, ob der Täter gefangen wird.

## 3.3 Schreibprozess des Skripts in Ren.py

### 3.3.1 Definieren der Charaktere

Um mit dem Skript zu beginnen, mussten als aller erstes der Charaktere und die Darstellungsweise ihres Textes definiert werden. Um im Dialog zwischen den verschiedenen Figuren zu erkennen wer gerade spricht, werden in einem Visual-Novel meist für jeden Charakter eigene Grafiken, TextBoxen und Text Formatierungen verwendet. Die Formatierung, welche in diesem Schritt zu definieren waren, stellen später im Spiel den Namen wie auch den Gesprochenen Text des Charakters dar. Um mehr Persönlichkeit ausdrücken zu können, können diese in Ren’ Py für jeden Charakter individuell angepasst werden. Unteranderem kann man so die Größe, Schriftart, Farbe, Transparenz und noch viele weitere Optionen von Namen und Text festlegen. Eine weitere Möglichkeit ist es ein TextBox-Design zu definieren, welches angezeigt wird, wenn dieser Charakter spricht. Das Speichern all dieser Einstellungen erfolgt durch das Erstellen einer Variabel, welche später im Code aufgerufen werden kann, um zu entschieden, wie der nachfolgende Text dargestellt wird.

In Abbildung 2 ist ein Beispiel zu sehen in welchem der Charakter Gabriel definiert wurde. Als erstes wurde wie gemäß den zuvor genannten Bedingungen eine Variabel gewählt, welche die Formatierungseinstellungen dieses spezifischen Charakters speichert. Durch das Abkürzen des Namens „Gabriel“ zu „g“ konnte der Code später effizienter geschrieben werden, blieb jedoch weiterhin gut verständlich.

Der erste Wert, der auf das Definieren des Charakters folgt, speichert den Namen. Dieser muss als String eingegeben werden und wird auf der TextBox angezeigt, wenn Gabriel spricht.

Mit Hilfe der nachfolgenden „Who“ und „What“ Tags wurden dann die Formatierungen, welche für den Text dieses Charakters zutreffen festgelegt. „Who“ Aussagen beziehen sich auf das Aussehen des Namens, während „What“ Aussagen das Aussehen des gesprochenen Textes verändern. Da Ren’ Py bereits einige Grundeinstellungen besitzt, welche bei nicht definierten Attributen eines Charakters verwendet werden, war das Ändern der Größe und des TextBox-Designs bei diesem Charakter nicht nötig.

A black background with colorful text

Description automatically generated

Abbildung 2: Definirung des Charakters Gabriel

A white rectangular object with black border

Description automatically generated

Abbildung 3: Textausgabe des Charakters Gabriel

Die Abbildung 3 zeigt, wie ein Text durch den nun definierten Charakter „Gabriel“ dargestellt wird.

In Destiny’s Reminder erhielt jeder Charakter eine eigene Schriftart und Farbe. Dies soll helfen die Persönlichkeit des jeweiligen Charakters noch klarer darzustellen, dient aber gleichzeitig auch dazu, dass ein Charakter schneller und einfacher erkannt werden kann, ohne den Namen lesen zu müssen.

Alle neuen, verwendeten Schriftarten wurden von „1001 Fonts“12 und „Dafont“13 heruntergeladen. Mit der Auswahl von „Minako-Regular“14, „Hey Comic15, „DraftingMono-Bold“16, „Typo\_Round\_Bold\_Demo“17, „KGRedHands“18, „gabriele-bad“19 und „texgyrecursor-bold“20 konnte ein breites Spektrum an handgeschriebenen und typografischen Schriftarten abgedeckt werden.

### 3.3.2 Schreiben der Szenen mittels Labels

Um das Navigieren des Codes und das Wechseln zwischen Szenen einfacher zu machen, arbeitet Ren’ Py mit einer eigenen „Label“ Funktion. Wie in Abbildung 4 zu sehen ist wird jeder neue beliebige Textabschnitt von einem Label Statement initiiert. Der Name dieses Labels kann frei gewählt werden und hilft danach diesen Teil des Codes wieder aufzurufen. Labels helfen das Skript zu strukturieren, sind jedoch auch nützlich, um von einer Szene zur nächsten zu springen.

Durch den Befehl „Jump“ kann zwischen den Labels beliebig hin und her gesprungen werden. Labels sind grundsätzlich nicht nötig um eine Geschichte mit Hilfe der Ren’ Py Engine zu schreiben, werden jedoch zu einem der wichtigsten Hilfsmittel, sobald interaktive Elemente eingebaut werden. So kann zwischen den verschiedenen Handlungs-Strängen gewechselt und wieder zu gemeinsamen Szene gesprungen werden, ohne den Text mehrfach im Skript vorhanden haben zu müssen.

A screenshot of a computer

Description automatically generated

Abbildung 4: Anwendung einer Label-Funktion

Innerhalb dieser Labels konnten nun die verschiedenen Szenen wie in einem gewöhnlichen Skript geschrieben werden. Jede neue Zeile enthält den Text der zu einem Zeitpunkt auf dem Bildschirm angezeigt wird und per Klick wird im Spiel dann zu nächsten gewechselt.

### 3.3.3 Einfügen Interaktiver Elemente

Menus:

Um wie zuvor genannt die Wahl einer Handlung zu ermöglichen, wurden Menus in das Skript eingebaut. Diese funktionieren grundsätzlich sehr ähnlich wie Labels, jedoch mit mehreren von dem Spieler/der Spielerin zu wählenden Optionen.

Durch das Aufrufen eines Menus wird im Spiel eine Auswahl von anklickbaren Optionen angezeigt. Diese können in beliebiger Menge und mit zuvor definiertem Text angezeigt werden. Um eine Option eines Menus zu generieren, wird dem Code ein String gegeben, welcher den Text enthält, der für diese Option angezeigt werden soll. Wird nun eine dieser Optionen angeklickt, springt das Programm zum Code, welcher unter dieser Option steht und springt nach dessen Ausführung, falls keine andere Anweisung gegeben wurde, wieder zum Code außerhalb des Menus. Auf Abbildungen 5 und 6 sind Beispiele eines Menus-Befehls und dessen generiertes Interface zu sehen.

Durch solche Menus können direkt kurze Texte ausgegeben werden oder mit Hilfe von Labels zu ganz neuen Handlungsstränge gesprungen werden. In Destiny’s Reminder wurden so alle der in Abbildung 1 aufgezeichneten Interaktiven Stationen in das Spiel eingebaut.

A black background with white text

Description automatically generated

Abbildung 5: Anwendung einer Menu-Funktion

Spieler-input:

Eine etwas kompliziertere weise den Spieler/die Spielerin in das Spielerlebnis mit einzubeziehen kam in der Form des Benennens einer Katze in einem der Abschnitte. Dies konnte mittels Python befehlen getan werden. Mit einem Input Statement kann das Spiel nach einem Text fragen und diesen in einer variabel speichern. Wie die Abbildung 7 zeigt, wurde dieser in Destiny’s Reminder in die Variabel „Catname“ untergebracht. Später im Spiel kann dann zum Beispiel bei der Definition des Namens der Katze auf diesen Text zugegriffen werden.

A black screen with orange text

Description automatically generated

Abbildung 7: Anwendung einer Spieler-input-Funktion

## 3.4 Illustrationsprozess der Spielgrafiken

Mit dem Plan der Handlung und den Anforderungen der 4 Bildklassen, welche während der Recherche-Phase definiert wurden, konnten die nötigen Illustrationen nun passend zur Geschichte angefertigt werden. Alle Illustrationen wurden auf einem iPad mit der Software Procreate21 erstellt.

### 3.4.1 Gestaltung der Charaktere und diverser Spielelemente

Charaktere:

Dank der dialogbasierten Spielweise eines Visual-Novels, sind es die Charaktere, welche die Geschichte zum Leben erwecken. Die Geschichte am Mittelpunkt dieser Arbeit, handelt von fünf sehr unterschiedlichen Personen. Um diesen fünf nicht nur durch ihre Sprechweisen, sondern auch durch ihr Aussehen voneinander unterscheiden zu können wurden für jeden der 5 Charaktere eigene Illustrationen angefertigt. Um die Emotionen der Figuren besser darzustellen und ein abwechslungsreicheres Spielerlebnis zu bieten werden in Visual-Novels oft mehrere verschiedene Illustrationen oder Animationen eines Charakters erstellt. Je nach Gefühl, welches gerade in einer bestimmten Aussage hervortritt, kann dies dann durch eine passende Grafik unterstrichen werden. Welche Emotion schlussendlich öfter gezeigt wird kann von Figur zu Figur variieren und somit helfen deren Persönlichkeit zu verstärken. Mit 4-5 Illustrationen pro Charakter konnten die wichtigsten Gefühlslagen und Gesichtsausdrücke abgedeckt werden.

Jede der 5 Figuren erhielt eine zentrale Farbe. Diese Farbe wurde sowohl für die Illustrationen wie auch bereits in Abschnitt 3.3.1 erwähnt für das Design des Textes verwendet. Die Abbildungen 8, 9 und 10 zeigen Beispiele für die verschiedenen Gefühlslagen, welche von einer Figur ausgedrückt werden können.

A cartoon of a person wearing glasses

Description automatically generated A person wearing glasses and a green sweater

Description automatically generated A cartoon of a person wearing glasses

Description automatically generated

Abbildung 8: Gabriel besorgt Abbildung 9: Gabriel verwirrt Abbildung 10: Gabriel glücklich

Weitere Spielelemente:

Gleich wie die Charaktere mussten auch die Illustrationen für jegliche andere Objekte wie zum Beispiel ein Handy oder ein Computer Bildschirm einzeln angefertigt und eingefügt werden. Der Malprozess blieb derselbe wie bei den Charakteren, mit dem einzigen unterschied, dass diese Objekte meist simpler und somit einfacher zu gestalten waren. Hier war es jedoch wichtig die verschiedenen Illustrationen auf die Hintergründe abzustimmen, auf welchen sie dargestellt werden würden. Als Folge wurden Grafiken wie beispielsweise die Achievement-screens, welche auf allen Hintergründen zu sehen sein würden, in einer sehr neutralen, meist schwarz weissen Palette gehalten. Andere Grafiken wie zum Beispiel der Computer, konnten hingegen gemäss der Palette des jeweiligen Hintergrunds coloriert werden.

### 3.4.2 Gestaltung der Hintergründe

Die Gestaltung der Hintergründe erfolgte in zwei Schritten. Da für die Geschichte relativ viele Hintergründe benötigt wurden, wurden für alle Illustrationen Bildern als Grundlage verwendet, welche entweder aus dem Internet heruntergeladen oder selbst gemacht wurden. Über diese Bilder konnten dann die verschiedenen Hintergrundillustrationen gemalt werden. Diese Art der Gestaltung vereinfachte das Zeichnen glaubhafter und räumlich wirkenden Szenen und minimierte den Zeitaufwand, da von bereits existierenden Strukturen ausgegangen werden konnte und nicht alle Räume von Hand gemalt werden mussten.

Für allgemeinere Orte, welche nur eine Illustration pro Raum beanspruchten, wie zum Beispiel Destiny’s Büro oder die Strasse, wurden Bilder aus dem Internet als Vorlage verwendet. Für Szenen aus dem Büro wurden Bilder von «Dreamstime»22 und «flickr»23 bezogen. Einer der Strassen-Hintergründe basiert auf eninem 3D Modell von Marteen Hof24. Um jedoch individuellere Orte wie beispielsweise die Cafeteria oder Destiny’s Wohnung von unterschiedlichen Blickwinkeln zeigen zu können, diente Sims425 als nützliches Hilfsmittel um die Räume zu bauen und die verschiedenen Ausschnitte zu fotografieren. In Abbildung 11 ist das Model Destiny’s Wohnung wie auch der fertige Hintergrund zu sehen.

A cartoon of a kitchen

Description automatically generated

Abbildung 11: Illustration von Destinys Wohnung mit Sims Model

Die Konturen der Aufnahmen wurden nachgezogen und allenfalls mit weiteren Objekten oder Änderungen ergänzt. Um ein einheitliches Aussehen für alle Hintergründe zu haben, wurde für das Färben der Bilder das Werkzeug „Verlaufsumsetzung“ verwendet. Dieses wird genutzt, um jedem hell-dunkel Wert neu auch eine bestimmte Farbe zu verleihen. Dies ermöglichte einen schnelleren Malprozess, da für die eher aufwändigeren Hintergründe nun nur noch auf die Helligkeit eines bestimmten Objekts geachtet werden musste. Zu den Illustrationen in Destiny’s Reminder wurden für die verschiedenen Orte eigene Verläufe erstellt und angewendet. In Abbildung 12 sind die einzelnen Verlaufsumsetzungen mit ihren zugehörigen Illustrationen zu sehen.

A screenshot of a video game

Description automatically generated

Abbildung 12: Verlaufsumsetzungen der Hintergründe

Um mögliche Unstimmigkeiten zwischen Hintergrund und Charakter zu vermeiden, wurden alle Szenen in einem sehr weichen Stil gestaltet. Es wurden hauptsächlich blasse Farben und schwache Kontraste verwendet. Die Farben einer Jeweiligen Umgebung wurden gewählt, um eine bestimmte Stimmung zu erzielen. Für Destiny’s Wohnung hieß das ein ruhiges Grün, für das Büro ein kaltes Blau und für die restlichen Orte ein blasses Rot. Die Hintergründe für die Szenen, welche nur mit der Entführung zu tun haben, wurden schwarz-weiß gelassen, um eine bedrückendere und leblosere Atmosphäre zu kreieren.

### 3.4.3 Gestaltung der Benutzeroberfläche und Icons

Für die Benutzeroberfläche des Spiels und den Desktop-Icons mussten natürlich auch noch eigene Grafiken angefertigt werden. Da der Fokus der Arbeit auf dem Spiel und seiner Geschichte lag, wurde für die Benutzeroberfläche überwiegend von den bereits von Ren‘ Py gegebenen Formen und Elementen ausgegangen. Mit wenigen Änderungen konnte bereits ein sehr individuelles Design erzielt werden. Die bereits Vorhandenen Grafiken wurden kopiert und in Procreate angepasst oder abgeändert und später mit der passenden Version ersetzt. Das Layout wurde ebenfalls mehrheitlich übernommen, mit leichten Anpassungen der Farben und dem Einfügen eigener Hintergründe. Die Schriftart des Menu-texts wurde Ebenfalls geändert. Das Wechseln der Grund-Schriftart von DejaVuSans26, der von Ren‘ Py als Standard genutzten Schriftart, zu CupCakes27 für in-game und Courier Prime Code28 für out-of-game Text brachten ebenfalls mehr Charakter in das Aussehen des Menus.

Zusammen mit einer Handvoll von Grund auf neuen Illustrationen für das Game-Icon und Startbildschirm wurde so eine stimmige, zum Spiel passende Benutzeroberfläche generiert.

## 3.5 Verknüpfen von Skript und Grafik

### 3.5.1 Einfügen eigener Grafiken und Schriftarten

Wie bereits zu beginn in Abschnitt 2.3 erwähnt erstellt Ren’ Py für jedes neue Projekt einen eigenen Ordner. Um eigene Illustrationen oder Fotos im Spiel verwenden zu können müssen diese lediglich sinnvoll benannt und im dafür vorgesehenen Ordner abgelegt werden. Für spielinterne Grafiken erstellt Ren’ py einen eigenen «Images» Ordner. Dank der bereits vordefinierten befehle von Ren’ Py können die Illustrationen der Charakter und Objekte also ganz einfach in diesen Ordner gelegt und später im Skript aufgerufen werden.

Die Grafiken der Benutzeroberfläche werden in einem separaten «GUI» Ordner abgelegt. Da für Destiny’s Reminder wie zuvor in Abschnitt 3.4.3 erwähnt die bereits gegebene Benutzeroberfläche lediglich angepasst wurde, mussten die dort vorhandenen Grafiken mit den selbst angefertigten oder bearbeiteten ersetzt werden. Hierbei war es wichtig ihren Namen nicht zu ändern, um sicher zu stellen, dass der Code später auch korrekt funktionieren würde.

Jegliche Schriftarten welche vom Spiel dargestellt werden sollen müssen ähnlich wie die Grafiken im Spielordner als «ttf» oder «otf» abgelegt werden.

### 3.5.2 Darstellen der Charakter und weiterer Spielelemente

Sobald die verschiedenen Illustrationen der Charaktere im korrekten Ordner hinterlegt waren, wurde das Aufrufen dieser sehr simpel. Wie bereits in Abschnitt 2.4 erwähnt, können einfache Bildelemente durch einen show-Befehl hervorgerufen und durch einen hide-Befehl wieder versteckt werden. Um nun zum richtigen Zeitpunkt die korrekte Emotion oder den korrekten Gegenstand darzustellen, musste erneut durch das Skript gegangen werden und zu jeder Aussage oder zu jedem Geschehnis die passenden Grafiken aufgerufen werden.

Ren’ Py gibt einem die Möglichkeit durch diesen show-Befehl viele Aspekte der Darstellungsweise anzupassen.

Um eine dynamische und glaubhafte Darstellung der Emotionen der Figuren darstellen zu können wurden diese zum Teil sehr oft geändert. Die Abbildung 13 zeigt eine solche Dialogstelle, in welcher das Wechseln der Emotionen gut zu sehen ist.

A screenshot of a computer screen

Description automatically generated

Abbildung 13: Wechseln zwischen Charakter-Grafiken

### 3.5.3 Effekte und Übergänge

Bla

# 4.Schluss

## 4.1 Beantwortung der Fragestellung

### 4.1.1 Wie schreibt man eine Interaktive Geschichte?

Eine Interaktive Geschichte unterscheidet sich von einer Linearen Geschichte nur in wenigen Elementen. Im Grunde sind Interaktive Geschichten lediglich eine Sammlung normaler Linearer Geschichten, welche individuell kombiniert und dann nacheinander gelesen werden. Um eine Interaktive Geschichte zu schreiben, muss man deshalb zu Beginn sehr ähnlich vorgehen wie bei einer Normalen Geschichte. Man plant eine zentrale Handlung und baut danach auf diese die verschiedenen Pfade auf. Die simpelste variante eine solche Geschichte zu schreiben, beginnt mit einem der Pfade bis hin zum ersten Ende. Von diesem spaltet man dann die weiteren verzweigenden Handlungen und Enden ab. Falls wie bei dieser Geschichte eines der Enden bevorzugt werden soll, eignet es sich von diesem Pfad auszugehen. Um für alle Pfade eine sinnreiche und Spannende Geschichte zu schreiben, erfordern Interaktive Geschichten natürlich etwas mehr Planung als Herkömmliche, doch schlussendlich beruhen beide auf denselben literarischen Grundlagen und nutzen dieselben Rhetorischen und Sprachlichen Mittel.

### 4.1.2 Wie setzt man eine Geschichte in eine Spielbare form um/Welche Methoden gibt es?

Um eine Geschichte in eine spielbare Form umzusetzen, gibt es viele Methoden. Nebst der hier gewählten form eines Videospiels, gibt es auch Bücher und TV-Serien welche Geschichten durch interaktive Elemente spielerisch umsetzten. Doch auch unter Videospielen gibt es eine Vielzahl an unterschiedlichen Wegen eine Geschichte zu erzählen. Zusätzlich zu Visual-Novels, deren Gameplay fast ausschließlich aus Dialog bestehet, gibt es auch Rollenspiele, RPGs genannt, welche den Spieler/der Spielerin direkt in die Geschichte eintauchen. Auch Spiele wie „Dungeons & Dragons“29, welche das Konzept eines Fantasy RPGs bereits in Brettspielform umsetzte, sind sehr beliebt. Alle diese Medien nutzen ihre individuellen Stärken und Möglichkeiten, um ansprechende und fesselnde interaktive Geschichten zu erzählen. Dank den vielen verschiedenen Methoden, welche verwendet werden können, eine Geschichte spielerisch darzustellen gibt es nicht den einen „richtigen“ Weg. Letzen Endes bestimmen die Präferenzen der Zielgruppe oder des Herstellers welche Form am geeignetsten ist.

### 4.1.3 Was braucht es, um einen Visual-Novel zu kreieren?

Das Herz jedes guten Visual-Novels ist eine spannende Geschichte. Sie bildet die Grundlage des ganzen Spiels und leitet den Spieler/die Spielerin durch das ganze Erlebnis. Um einen simplen Visual-Novel zu erstellen, braucht es grundsätzlich nur ein Skript, welches gesprochen wird, einfache Grafiken und ein Programm, welches diese Dinge verknüpfen und darstellen kann. Für diese Arbeit wurde deshalb die Engine Ren‘ Py gewählt da sie genau für diesen Zweck erstellt wurde und bereits viele der nötigen Elemente eines Visual-Novels liefert. In den von größeren Studios gemachten Spielen sieht man meist eine hohe Zahl an Orten und Charakteren, welche die Geschichte auskleiden. Dinge wie wechselnde Hintergründe, verschiedene Gesichtsausdrücke oder Übergänge sind jedoch kein muss. Sie können als Stilmittel eingesetzt werden sind und helfen ein ansprechendes Spielerlebnis zu bieten, sind jedoch nicht in allen Fällen nötig. Auch simple Grafiken können eine gute Geschichte erfolgreich ergänzen. Auch Interaktivität ist keine Bedingung, um einen guten Visual-Novel zu kreieren. Es gibt viele Linearer Geschichten, welche durch dieses Medium dargestellt wurden. Entscheidet man sich jedoch für eine Version mit verschiedenen Pfaden ist es wichtig diese so zu gestalten, dass die Geschichte für alle Pfade sinn ergibt.

### 4.1.4 Wie verstärken visuelle Elemente den Eindruck einer Geschichte?

Um diese Frage beantworten zu können war es wichtig den Unterschied zwischen einem Drama, beziehungsweise einem Dialog basiertem Text wie dem eines Visual-Novels, zu anderen Texten zu verstehen. Texte der Epik oder Lyrik sind geschrieben, im Wissen, das der Text das vollendete werk darstellt. Sie beinhalten malerische Beschreibungen der Figuren, Orte und Atmosphären um dem Leser/der Leserin die Möglichkeit zu geben sich die beschriebene Szene selbst vorzustellen und die Emotionen, welche präsent sind aufzufassen. Texte wie diese lassen mehr Raum für Interpretation und können teils auch gänzlich ohne direkte rede erfolgreich Situationen darstellen, in welchen eine Konversation stattfindet. Hier findet sich auch einen der größten Unterschiede zu einem Skript. Ein Skript oder ein Drama kann zwar als reiner Text gelesen und verstanden werden, wird jedoch grundsätzlich geschrieben, um vorgeführt oder aufgenommen zu werden.

Die Bilder und Grafiken werden eingesetzt, um dem gesprochenen Text mehr Kontext zu verleihen und die Emotionen der Figuren darzustellen. Wie der von einer bestimmten Figur gesprochene Text zu interpretieren ist kann durch die dargestellte Grafik beeinflusst werden. Aussagen können Emotional geladen werden oder gar eine komplett andere Bedeutung erhalten je nach Bild welches gleichzeitig angezeigt wird. Auch die Hintergründe verleihen den Szenen weiteren Kontext ohne zusätzliche Texte verwenden zu müssen.

Was in einem Drama oder Film gesprochen wird, wird in einem Visual-Novel hingegen gelesen. und die Wirkung einer Geschichte zu verstärken. Bücher und andere rein Textbasierte Medien nutzen oft eingesetzt werden, um dem gesprochenen Text mehr Kontext zu verleihen.

### 4.1.5 Wie kann ich meinen Code strukturiert notieren (Clean Code)?

Die genutzte Programmier-Engine Ren‘ Py gibt dem Nutzer/der Nutzerin bereits eine grobe Struktur vor. Der Code ist auf 4 Dateien verteilt. 3 der 4 Dateien regulieren und definieren Wie das Spiel und dessen Benutzerinterface angezeigt und abgespielt wird. Diese können trozdem beliebig geändert werden, um das Spiel dem Inhalt anzupassen, beinhalten jedoch auch wichtigen code welcher nötig ist um das Spiel starten zu können. Ren‘ Py beschreibt deshalb bereits welche Aspekte des Codes zu bearbeiten sind und welche nicht geändert werden dürfen. Da sich die Struktur des in diesen Dateien vorhandenen Codes durch das Bearbeiten nicht wesentlich verändert konnte diese gelassen werden, wie sie war.

Die letzte Datei beinhaltet das Skript und ist vom Nutzer/der Nutzerin frei zu bearbeiten. Bei normalem Programmieren ist das Ziel den Code so kurz und effizient wie möglich zu halten. Dies sorgt dafür, dass das geschriebene Programm übersichtlich bleibt und wenig Speicherplatz beansprucht. Damit Ren‘ Py jedoch ein Skript geschrieben wird sieht die Datenstruktur dieser Dokumente etwas anders aus. Wie bei herkömmlichem Programmieren, müssen in Ren‘ Py jegliche Befehle und Variablen, somit auch die Charaktere, vorab definiert werden. Das Programm arbeitet, wie bereits in Abschnitt 3.3.2 erwähnt mit einer Label-Funktion. Dies bedeutet, dass die verschiedenen Textstellen beliebig angeordnet werden können. Wie die Szenen im Dokument sortiert sind, bleibt also frei zu wählen. Bis ein Jump-Befehl angetroffen wird, wird der code von oben nach unten ausgespielt. Was bedeutet, dass auch durch korrektes Anordnen von einem Label zum nächsten gewechselt werden kann.

Da diese Geschichte einen klaren Zeitlichen Verlauf besitzt machte es Sinn die Szenen in Chronologischer Reihenfolge anzuordnen. Die verschiedenen Versionen von Szenen, basierend auf wie gewählt wurde, wurden untereinandergestellt. Ebenfalls wurden Szenen, welche in mehreren Routen vorkommen, mit eigenen Labels versehen, um aus den verschiedenen Pfaden auf sie zugreifen zu können und so mehrfach vorhandenen Text zu vermeiden. Die Chronologische Anordnung kombiniert mit

# 5. Reflexion

## 5.1 Inhalt

Bla.

## 5.2 Methoden

Bla.

# 6. Quellenverzeichnis

1. Microsoft Corporation (2023), GitHub (Version 3.10.2)
2. Wikipedia: <https://de.wikipedia.org/wiki/GitHub> (Abgerufen am 8.10.23)
3. Packard E., Die Insel der 1000 Gefahren, (Ravensburger Verlag, 2007)
4. David Slade, (Netflix, 2018), “Bandersnatch”, Black Mirror
5. P-Studio/Atlus/Sega (2016). Persona5
6. Team Salvato (2017), Doki Doki Literature Club
7. Emmarie.sites.gettysbvurg :<https://emmarie.sites.gettysburg.edu/visual-novels/> (Abgerufen 9.10.23)
8. Tom "PyTom" Rothamel (2023), Ren’ Py (Version 8.0.3)
9. Renpy: <https://www.renpy.org> (Abgerufen 9.10.23)
10. Learnattack: <https://learnattack.de/schuelerlexikon/deutsch/grundformen-des-dramas> (Abgerufen 9.10.23)
11. Draw.io: <https://app.diagrams.net> (Abgerufen 9.10.23)
12. 1001Fonts: <https://www.1001fonts.com> (Abgerufen 9.10.23)
13. Dafont: <https://www.dafont.com> (Abgerufen 9.10.23)
14. Minako: <https://www.dafont.com/minako.font> (Abgerufen 9.10.23)
15. Hey Comic: <https://www.dafont.com/hey-comic.font> (Abgerufen 9.10.23)
16. Drafring\* Mono: <https://www.1001fonts.com/drafting-mono-font.html> (Abgerufen 9.10.23)
17. Typo Round: <https://www.1001fonts.com/typo-round-font.html> (Abgerufen 9.10.23)
18. KGRedHands: <https://www.dafont.com/kg-red-hands.font> (Abgerufen 9.10.23)
19. Gabriele-Bad: <https://www.1001fonts.com/gabriele-font.html> (Abgerufen 9.10.23)
20. TeX Gyre Cursor: <https://www.1001fonts.com/tex-gyre-cursor-font.html> (Abgerufen 9.10.23)
21. Savage Interactive (2023), Procreate (Version 5.3)
22. Dreamstime: <https://www.dreamstime.com> (Abgerufen 9.10.23)
23. Flickr: <https://www.flickr.com/photos/markjsebastian/1264424156/> (Abgerufen 9.10.23)
24. Marteen Hof, Artstation: <https://maartenhof.artstation.com/projects/dOGYNK> (Abgerufen 9.10.23)
25. Maxis (2014), The Sims4 (Version 1.101.290.1230)
26. Deja Vu Sans: <https://www.1001fonts.com/dejavu-sans-font.html> (Abgerufen 9.10.23)
27. CupCakes: <https://www.dafont.com/cupcakes.font> (Abgerufen 9.10.23)
28. Courier Prime: <https://www.1001fonts.com/courier-prime-font.html> (Abgerufen 9.10.23)
29. Gary Gygax/Dave Arneson (1974), Dungeons & Dragons

# 7. Anhang

Bla